JAVIER DUARTE

ASPEN WINTER CONFERENCE

MARCH 28, 2023

ML FOR TRIGGERING

I. INTRO & MOTIVATION

II. COMPRESSION

III. HARDWARE

IV. APPLICATIONS

COMPUTING HARDWARE

Image: [Microsoft]

COMPUTING HARDWARE

Image: Microsoft

COMPUTING HARDWARE

Image: [Microsoft]

COMPUTING HARDWARE

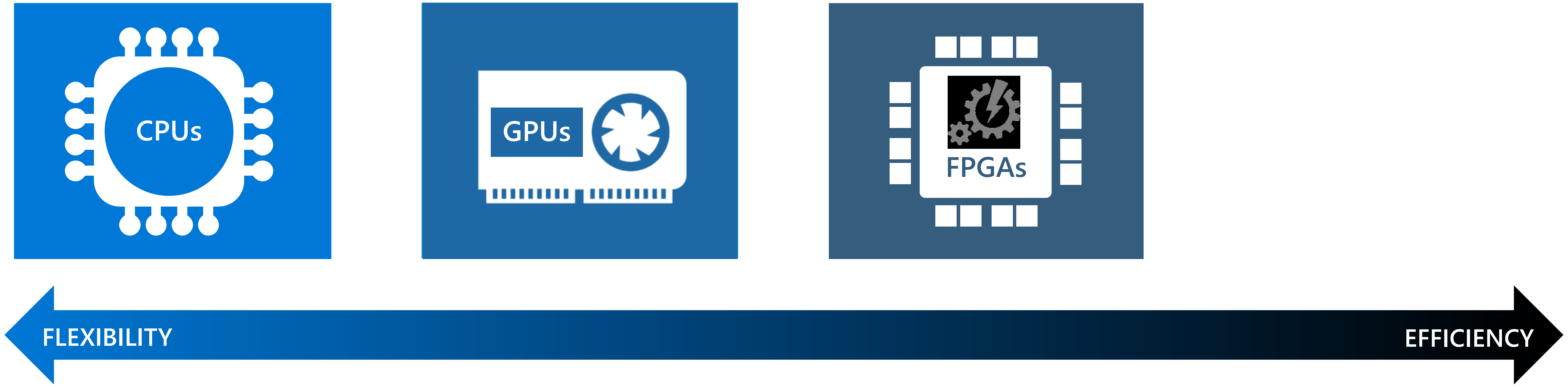


Image: [Microsoft]

COMPUTING HARDWARE

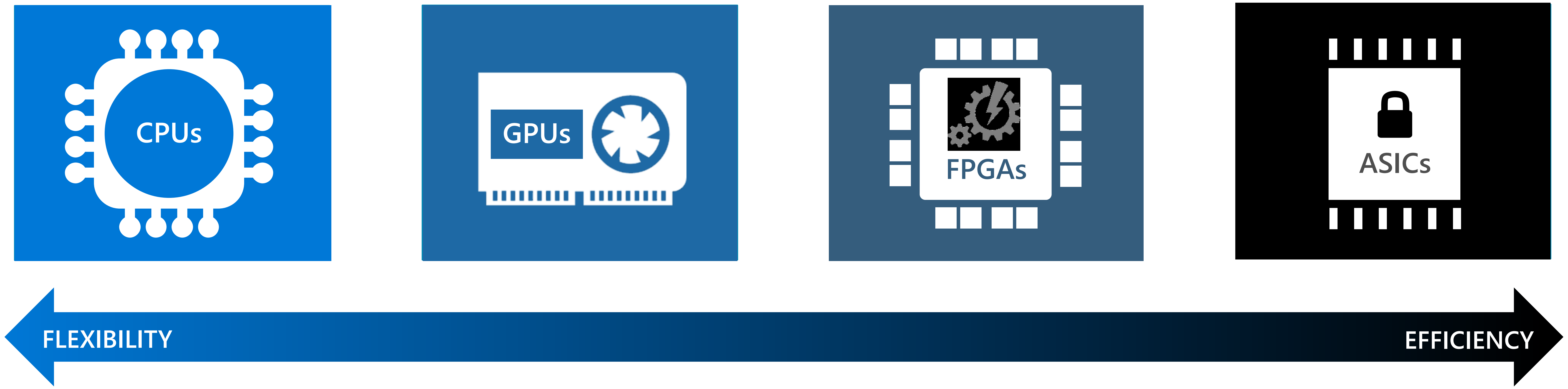


Image: [Microsoft]

COMPUTING HARDWARE

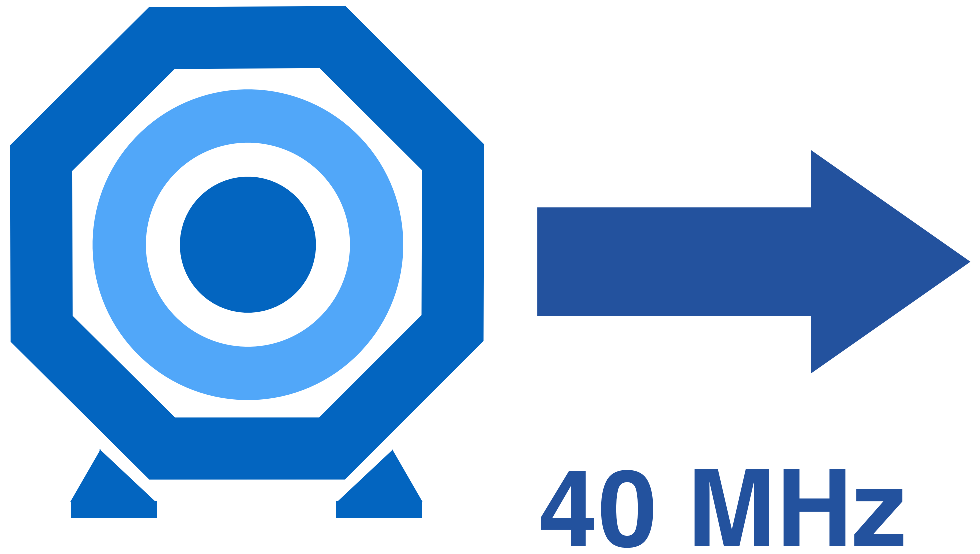
First stage of
LHC *trigger*

Image: [Microsoft]

HL-LHC EVENT PROCESSING

Compute Latency

1 ns 1 μs 1 ms 1 s



40 MHz

ASICs

Challenges:

Each collision produces $O(10^3)$ particles

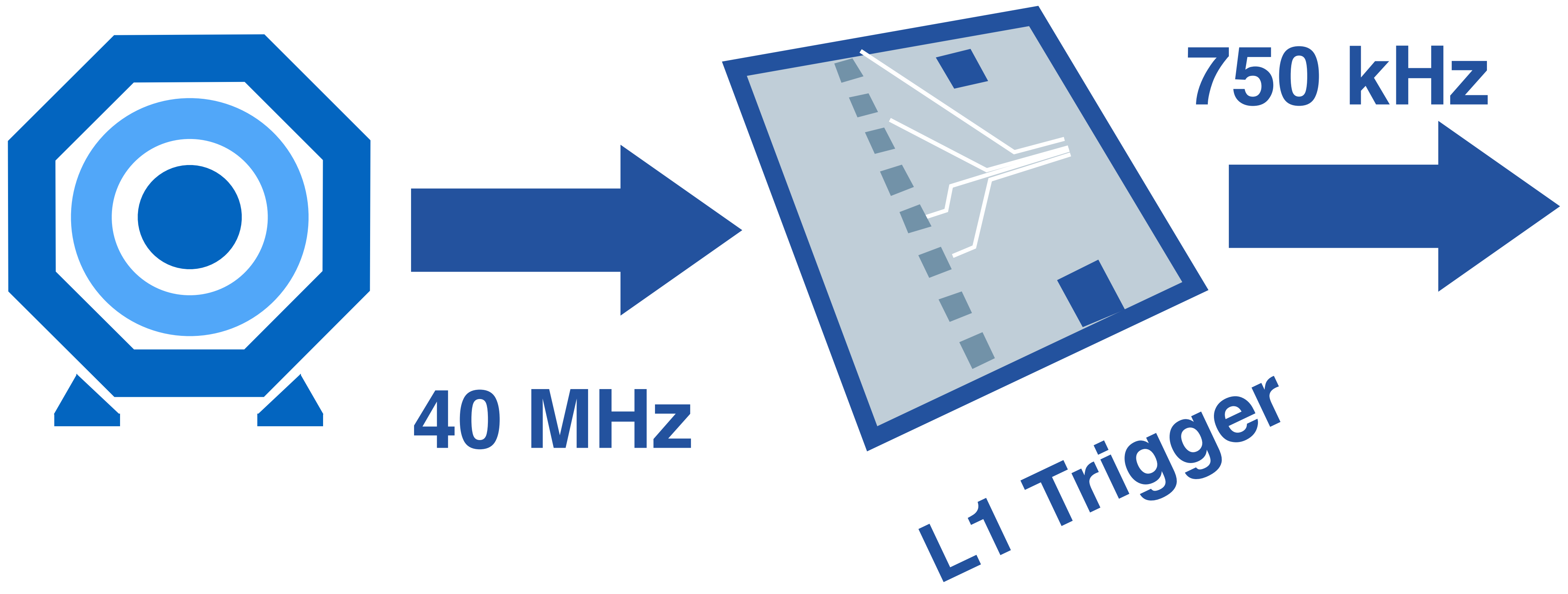
The detectors have $O(10^8)$ sensors

Extreme data rates of O(100 TB/s)

HL-LHC EVENT PROCESSING

Compute Latency

1 ns — 1 μs — 1 ms — 1 s



ASICs

FPGAs

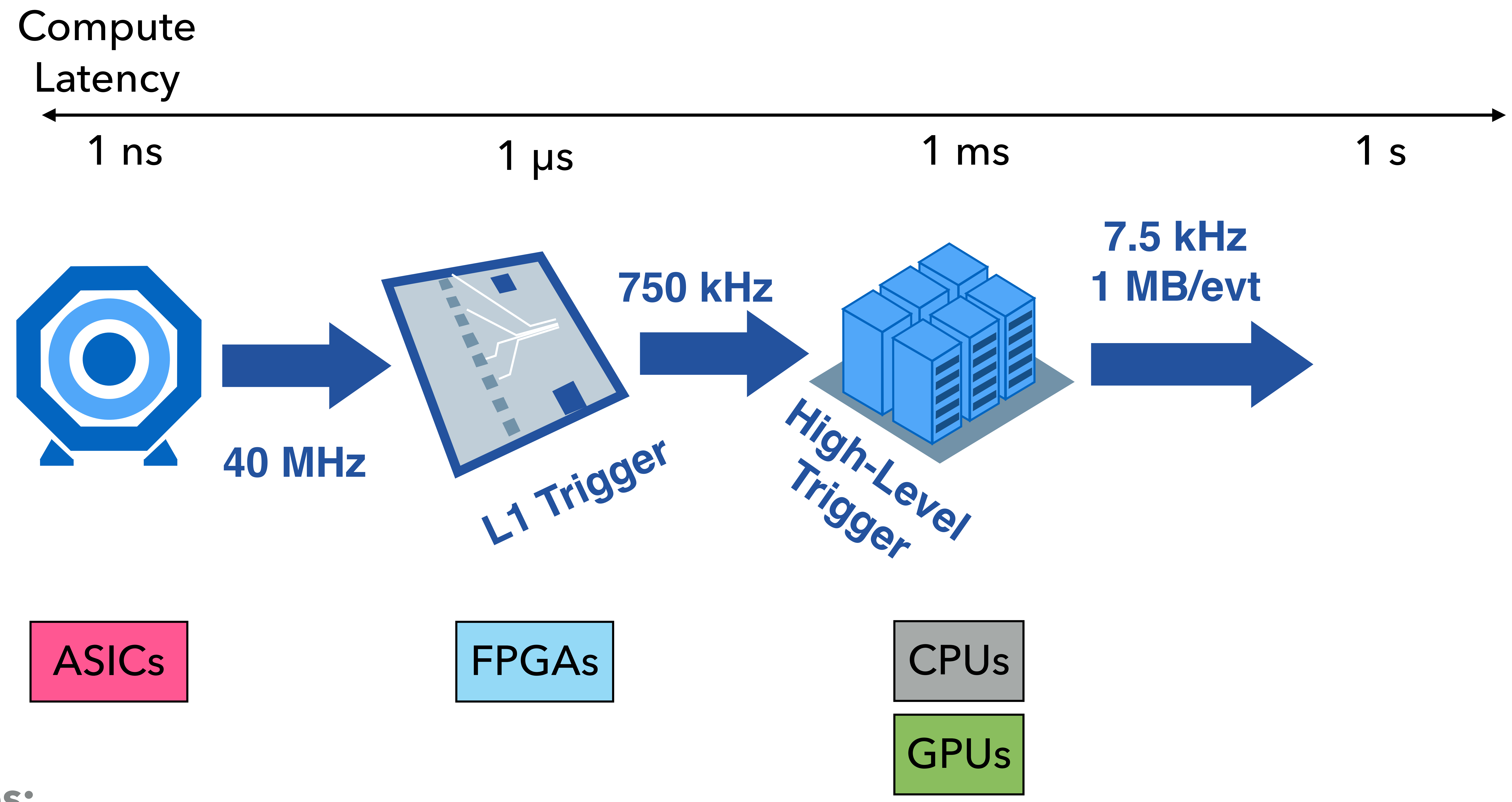
Challenges:

Each collision produces $O(10^3)$ particles

The detectors have $O(10^8)$ sensors

Extreme data rates of O(100 TB/s)

HL-LHC EVENT PROCESSING



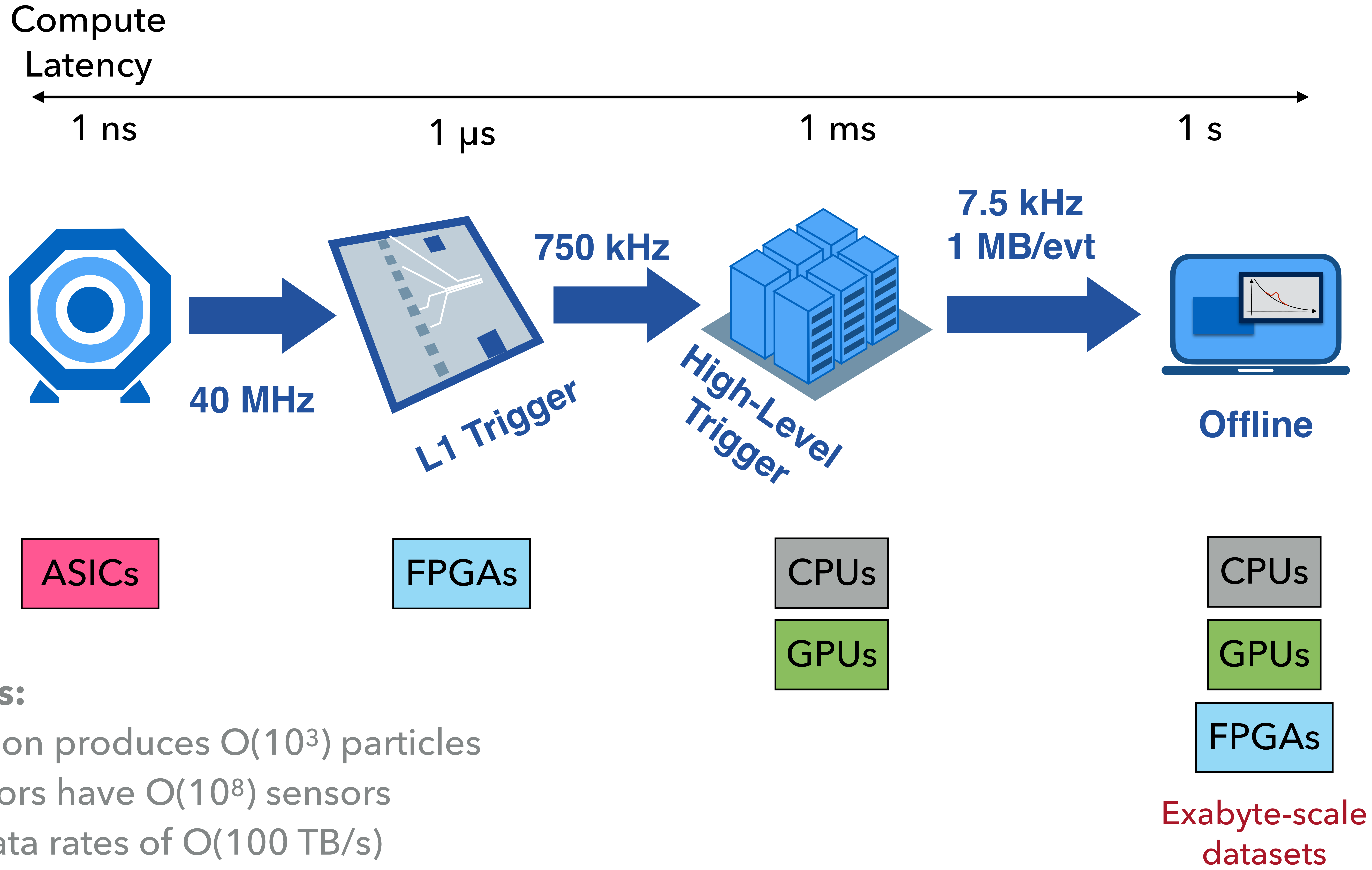
Challenges:

Each collision produces O(10^3) particles

The detectors have O(10^8) sensors

Extreme data rates of O(100 TB/s)

HL-LHC EVENT PROCESSING



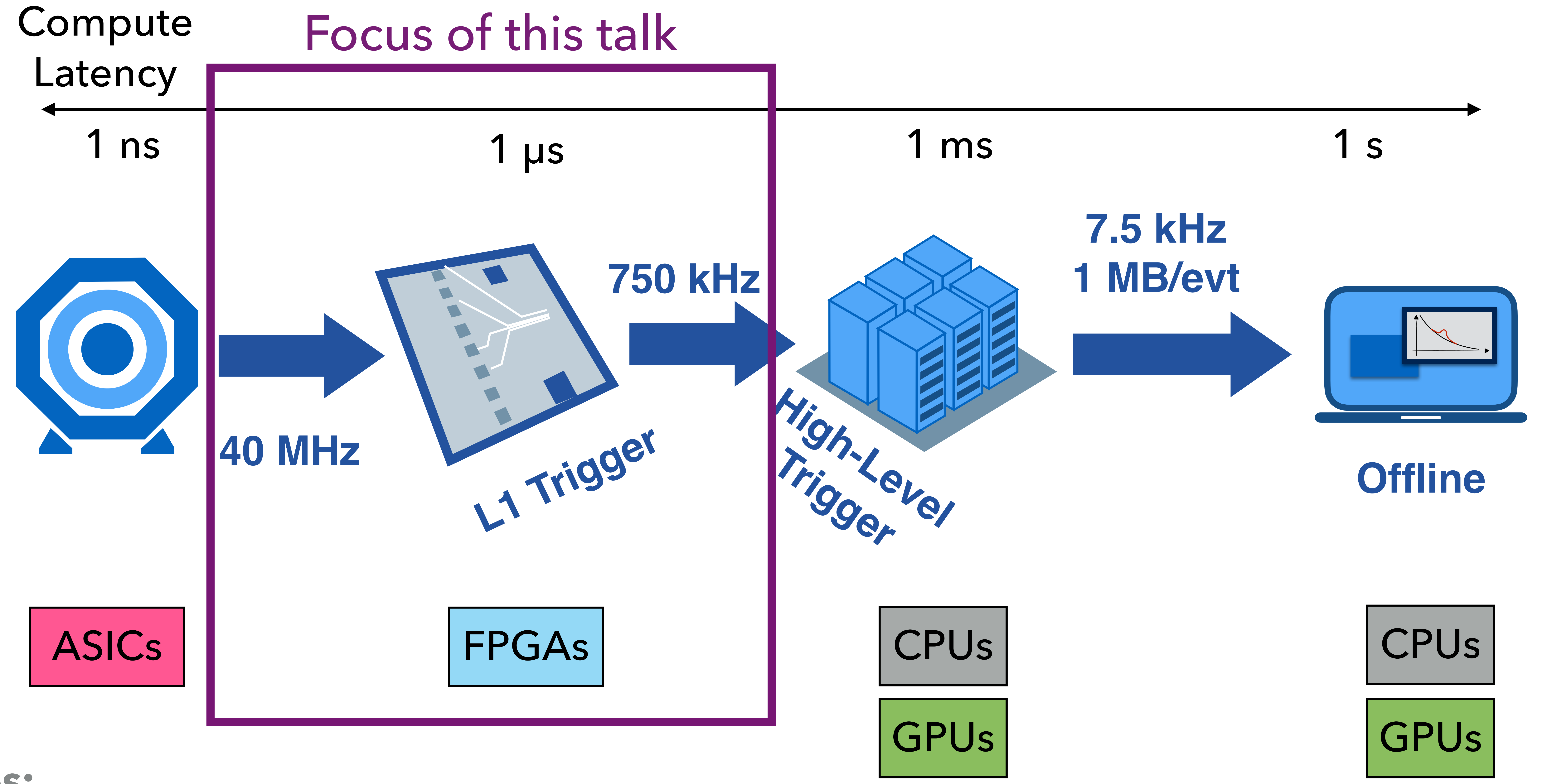
Challenges:

Each collision produces $O(10^3)$ particles

The detectors have $O(10^8)$ sensors

Extreme data rates of $O(100\ \text{TB/s})$

HL-LHC EVENT PROCESSING



Challenges:

Each collision produces $O(10^3)$ particles

The detectors have $O(10^8)$ sensors

Extreme data rates of O(100 TB/s)

SIMPLIFIED HL-LHC L1 TRIGGER MENU

Thresholds set by
backgrounds,
limited resolution
@ L1, and rate
budget

CMS-TDR-021

SIMPLIFIED HL-LHC L1 TRIGGER MENU

Trigger	Threshold [GeV]

Thresholds set by backgrounds, limited resolution @ L1, and rate budget

CMS-TDR-021

SIMPLIFIED HL-LHC L1 TRIGGER MENU

- Single/double/triple muons/electrons

Trigger	Threshold [GeV]
1 μ	22
2 μ	15, 7
3 μ	5, 3, 3
1 e	36
2 e	25, 12

Thresholds set by backgrounds, limited resolution @ L1, and rate budget

CMS-TDR-021

SIMPLIFIED HL-LHC L1 TRIGGER MENU

- Single/double/triple muons/electrons
- Photons

Trigger	Threshold [GeV]
1 μ	22
2 μ	15, 7
3 μ	5, 3, 3
1 e	36
2 e	25, 12
1 γ	36
2 γ	22, 12

Thresholds set by backgrounds, limited resolution @ L1, and rate budget

CMS-TDR-021

SIMPLIFIED HL-LHC L1 TRIGGER MENU

- Single/double/triple muons/electrons
- Photons
- Taus

Trigger	Threshold [GeV]
1 μ	22
2 μ	15, 7
3 μ	5, 3, 3
1 e	36
2 e	25, 12
1 γ	36
2 γ	22, 12
1 τ	150
2 τ	90, 90

Thresholds set by backgrounds, limited resolution @ L1, and rate budget

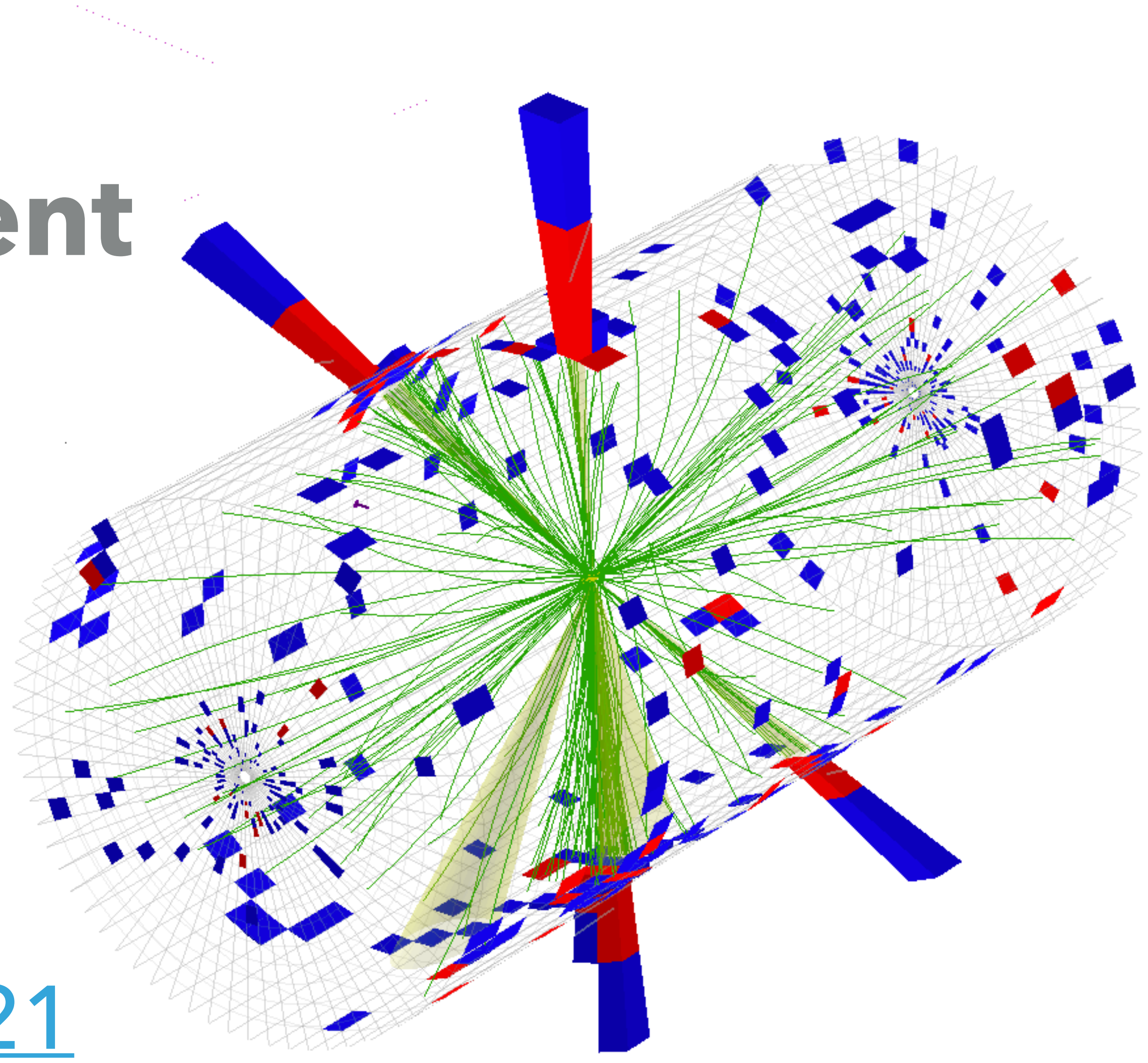
CMS-TDR-021

SIMPLIFIED HL-LHC L1 TRIGGER MENU

- Single/double/triple muons/electrons
- Photons
- Taus
- Hadronic

Trigger	Threshold [GeV]
1 μ	22
2 μ	15, 7
3 μ	5, 3, 3
1 e	36
2 e	25, 12
1 γ	36
2 γ	22, 12
1 τ	150
2 τ	90, 90
1 jet	180
2 jet	112, 112
H_T	450
4 jet + H_T	75, 55, 40, 40, 400

4-jet event



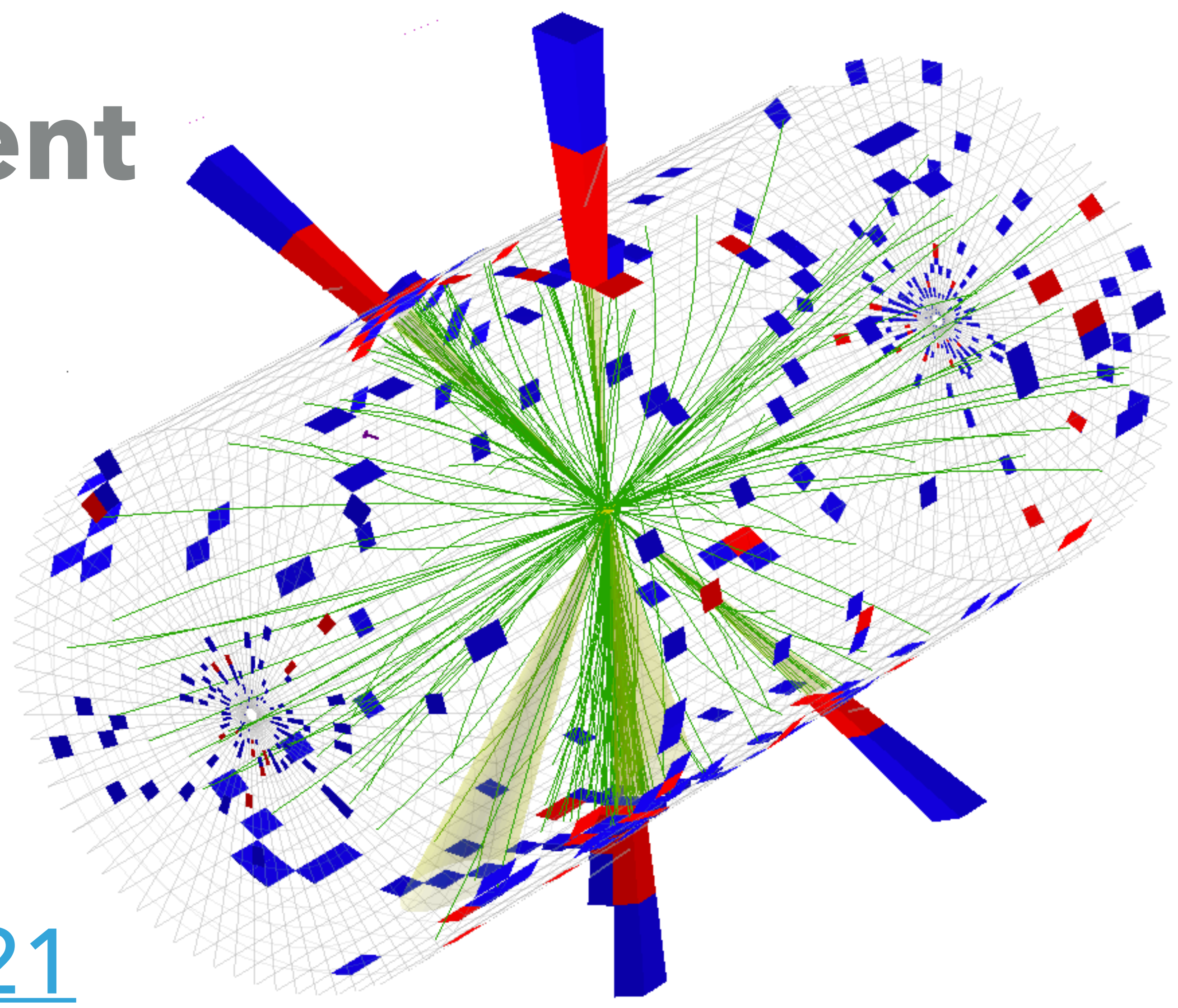
Thresholds set by backgrounds, limited resolution @ L1, and rate budget

CMS-TDR-021

SIMPLIFIED HL-LHC L1 TRIGGER MENU

- Single/double/triple muons/electrons
- Photons
- Taus
- Hadronic
- Missing transverse energy

4-jet event



CMS-TDR-021

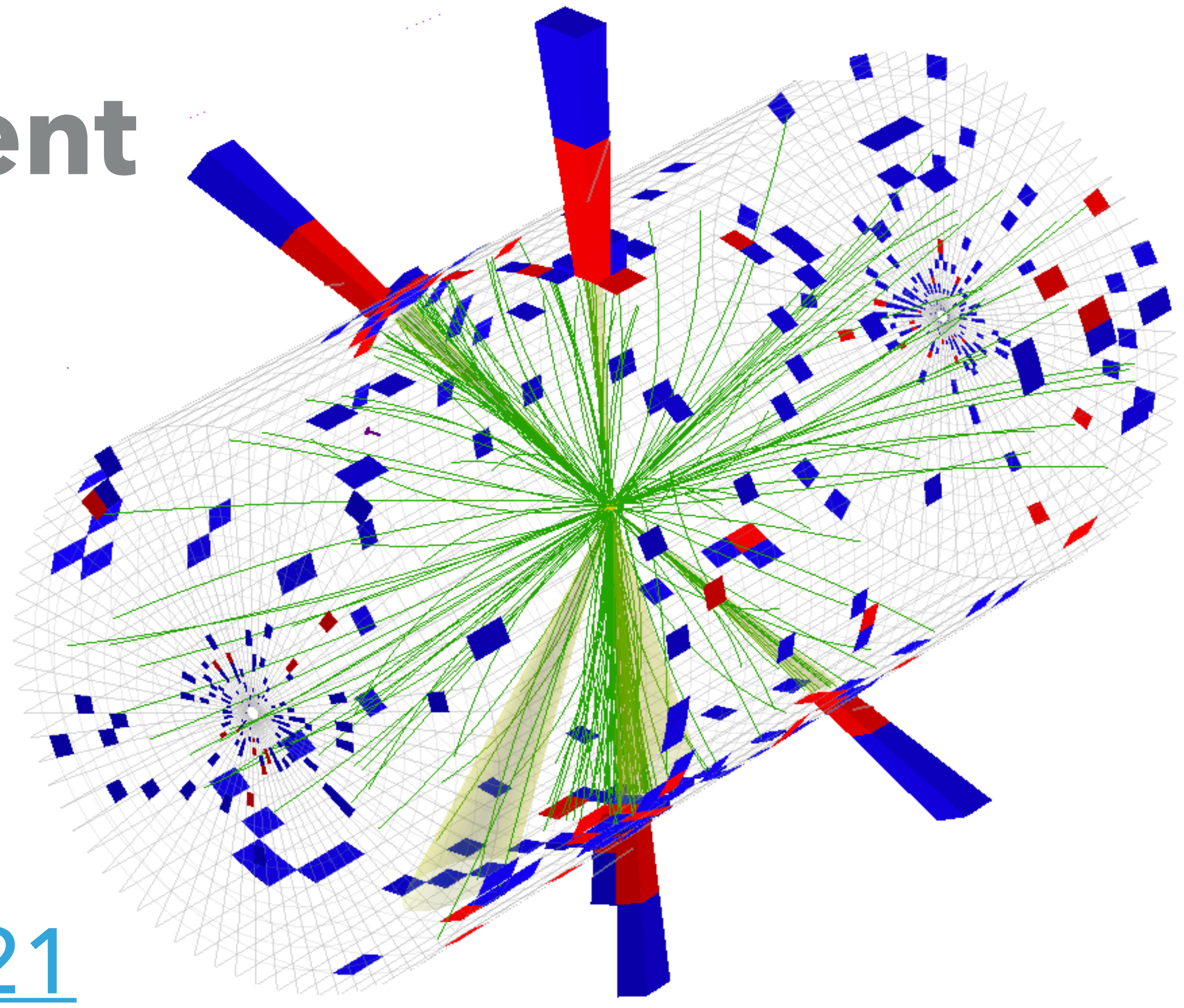
Thresholds set by backgrounds, limited resolution @ L1, and rate budget

Trigger	Threshold [GeV]
1 μ	22
2 μ	15, 7
3 μ	5, 3, 3
1 e	36
2 e	25, 12
1 γ	36
2 γ	22, 12
1 τ	150
2 τ	90, 90
1 jet	180
2 jet	112, 112
H_T	450
4 jet + H_T	75, 55, 40, 40, 400
p_T^{miss}	200

SIMPLIFIED HL-LHC L1 TRIGGER MENU

- Single/double/triple muons/electrons
- Photons
- Taus
- Hadronic
- Missing transverse energy
- "Cross" triggers (not shown)

4-jet event



CMS-TDR-021

Thresholds set by backgrounds, limited resolution @ L1, and rate budget

Trigger	Threshold [GeV]
1 μ	22
2 μ	15, 7
3 μ	5, 3, 3
1 e	36
2 e	25, 12
1 γ	36
2 γ	22, 12
1 τ	150
2 τ	90, 90
1 jet	180
2 jet	112, 112
H_T	450
4 jet + H_T	75, 55, 40, 40, 400
p_T^{miss}	200

WHAT COULD WE BE MISSING?

WHAT COULD WE BE MISSING?

- How can we trigger on more complex low-energy hadronic signatures? Long-lived/displaced particles?

SUEP

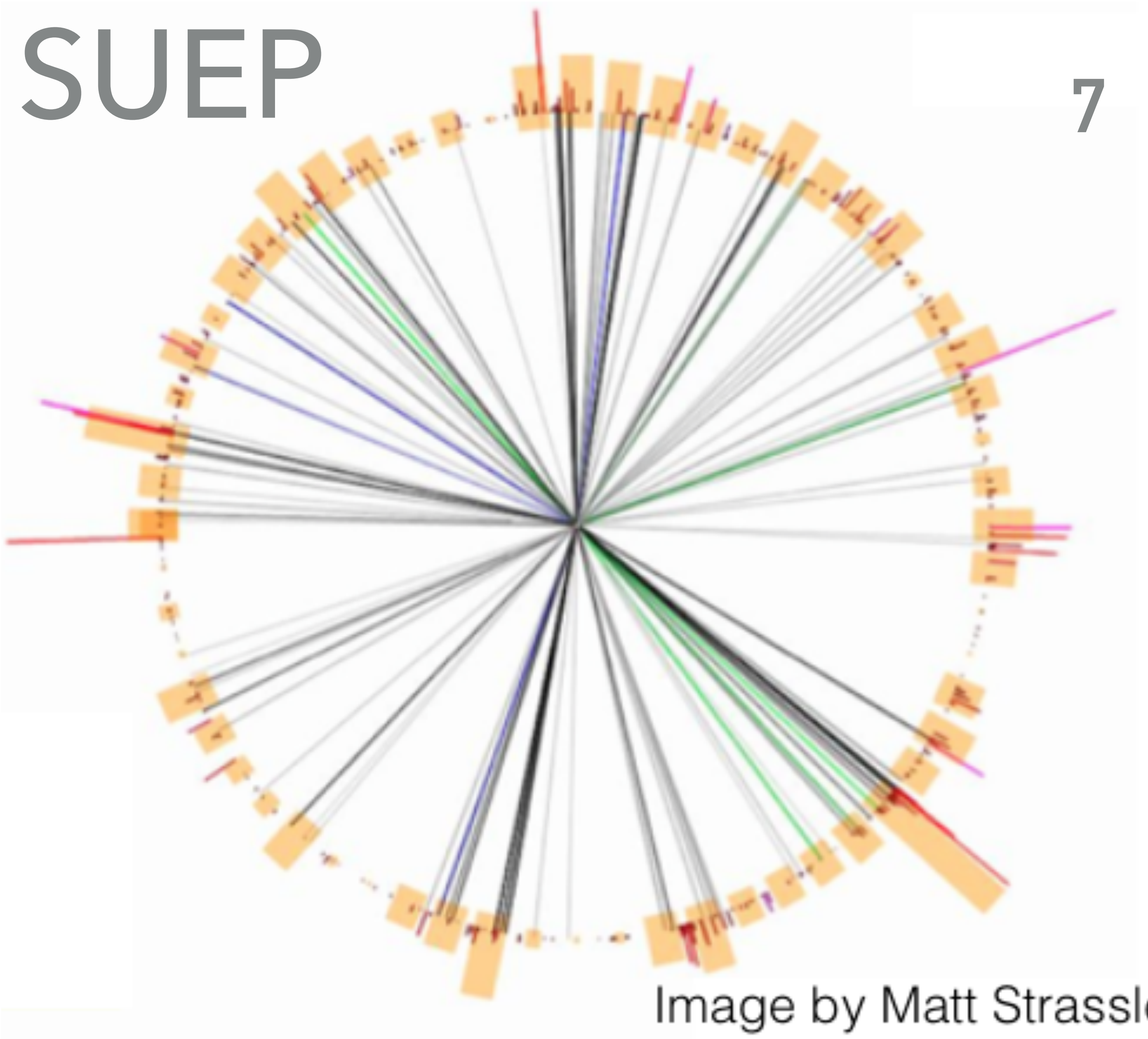


Image by Matt Strassler

WHAT COULD WE BE MISSING?

- How can we trigger on more complex low-energy hadronic signatures? Long-lived/displaced particles?
- What if we don't know exactly what to look for?

SUEP

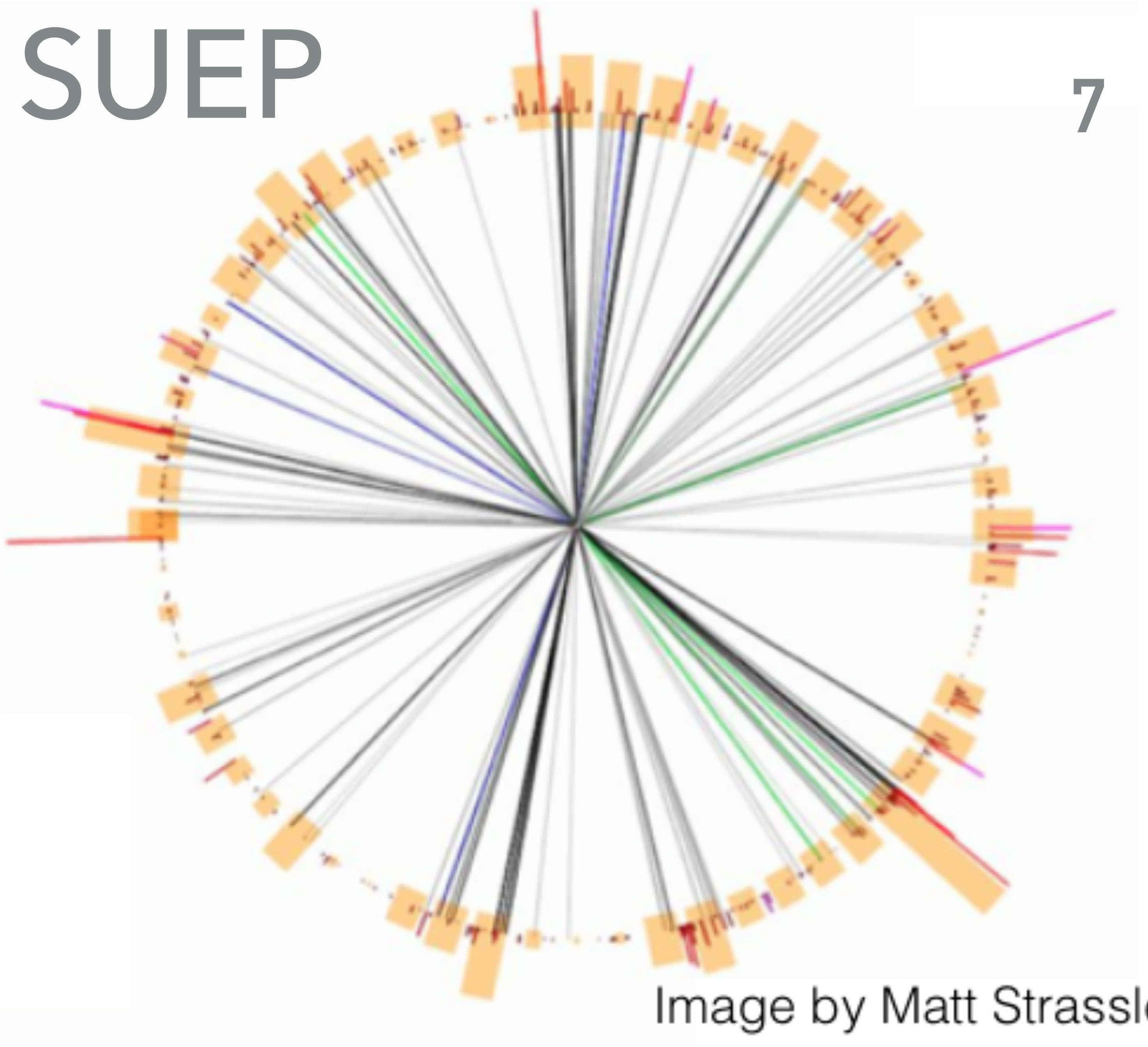
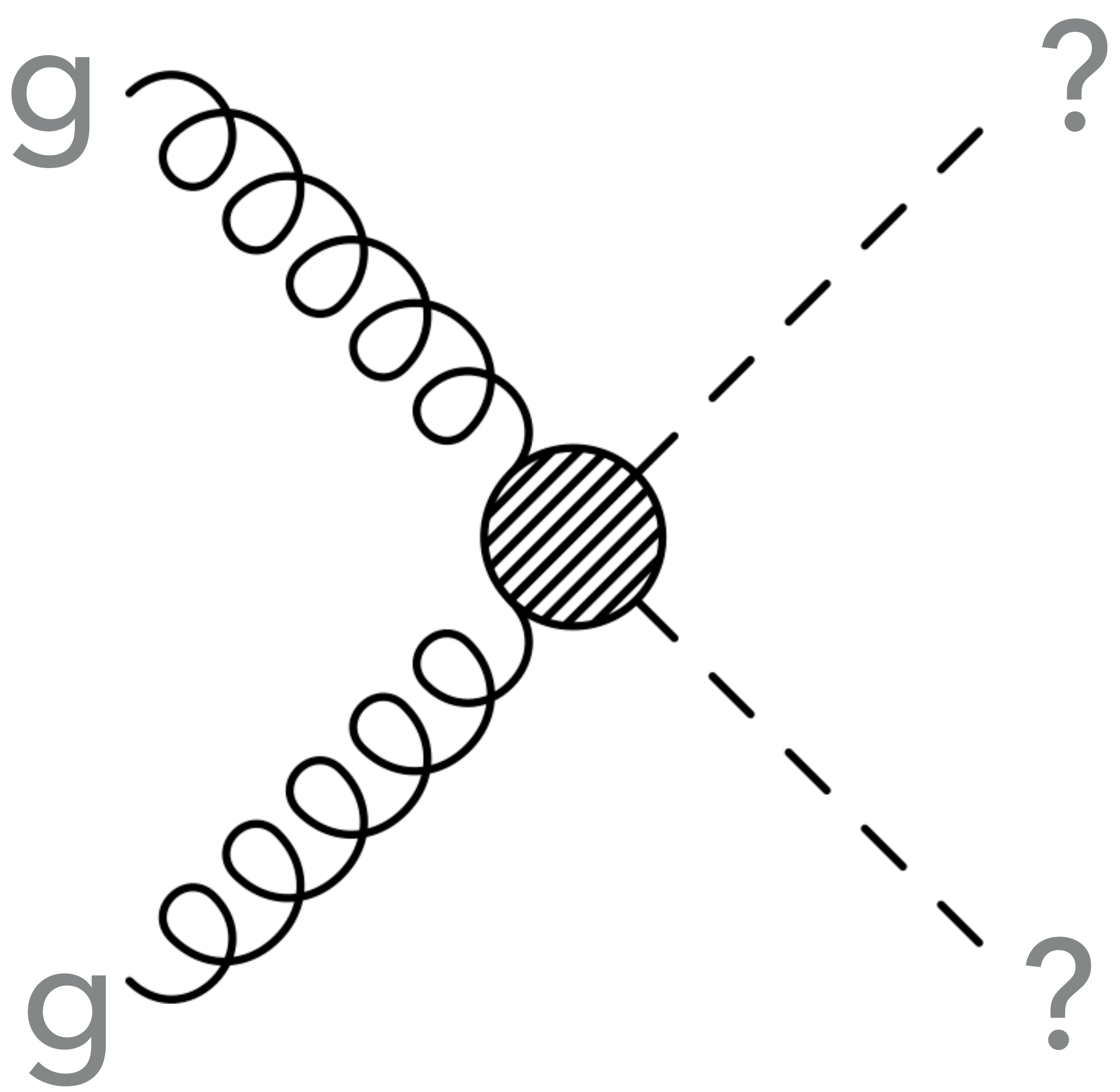


Image by Matt Strassler



WHAT COULD WE BE MISSING?

- How can we trigger on more complex low-energy hadronic signatures? Long-lived/displaced particles?
- What if we don't know exactly what to look for?
- What if our signatures require complex multivariate algorithms (e.g. b tagging)?

SUEP

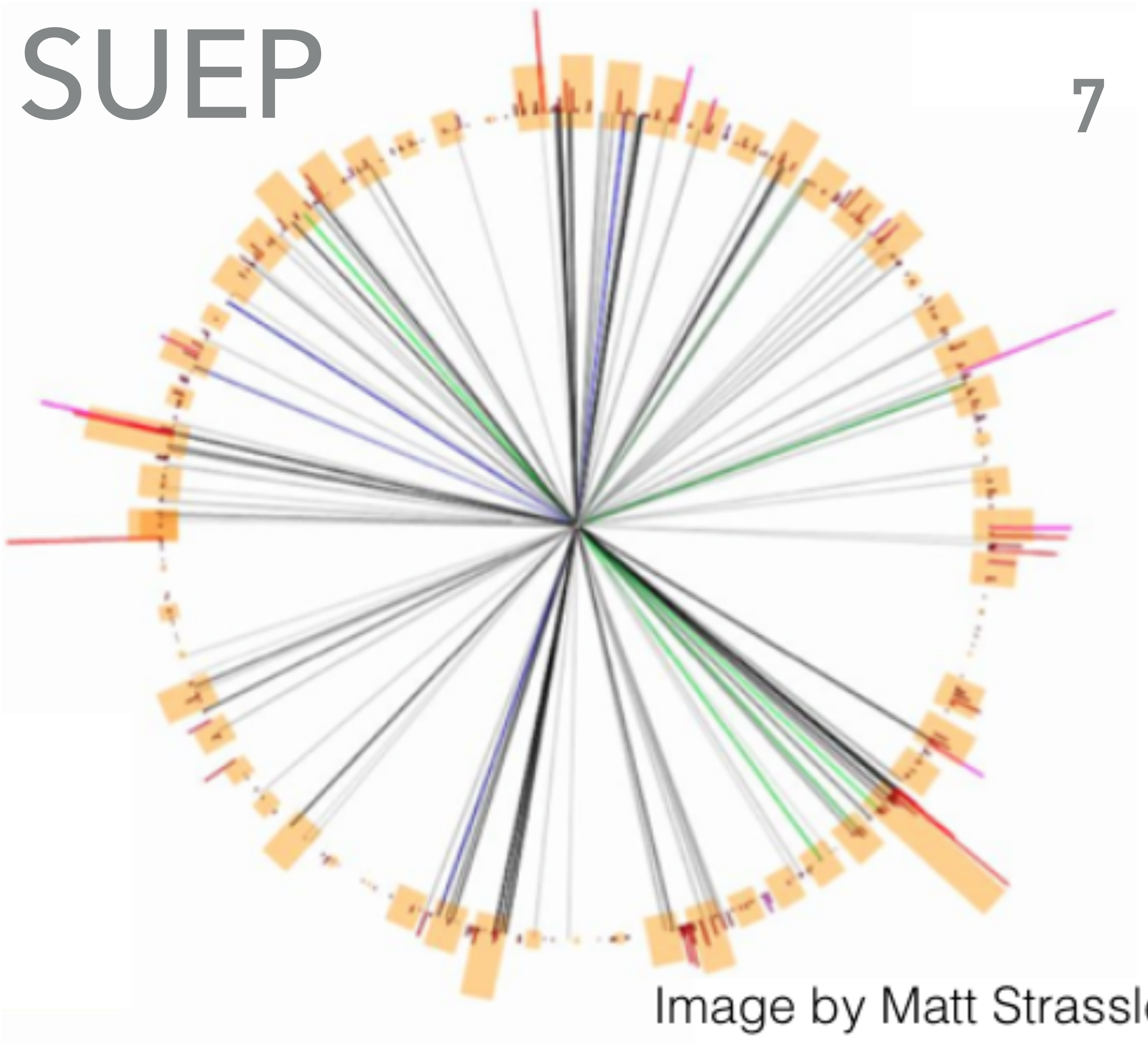
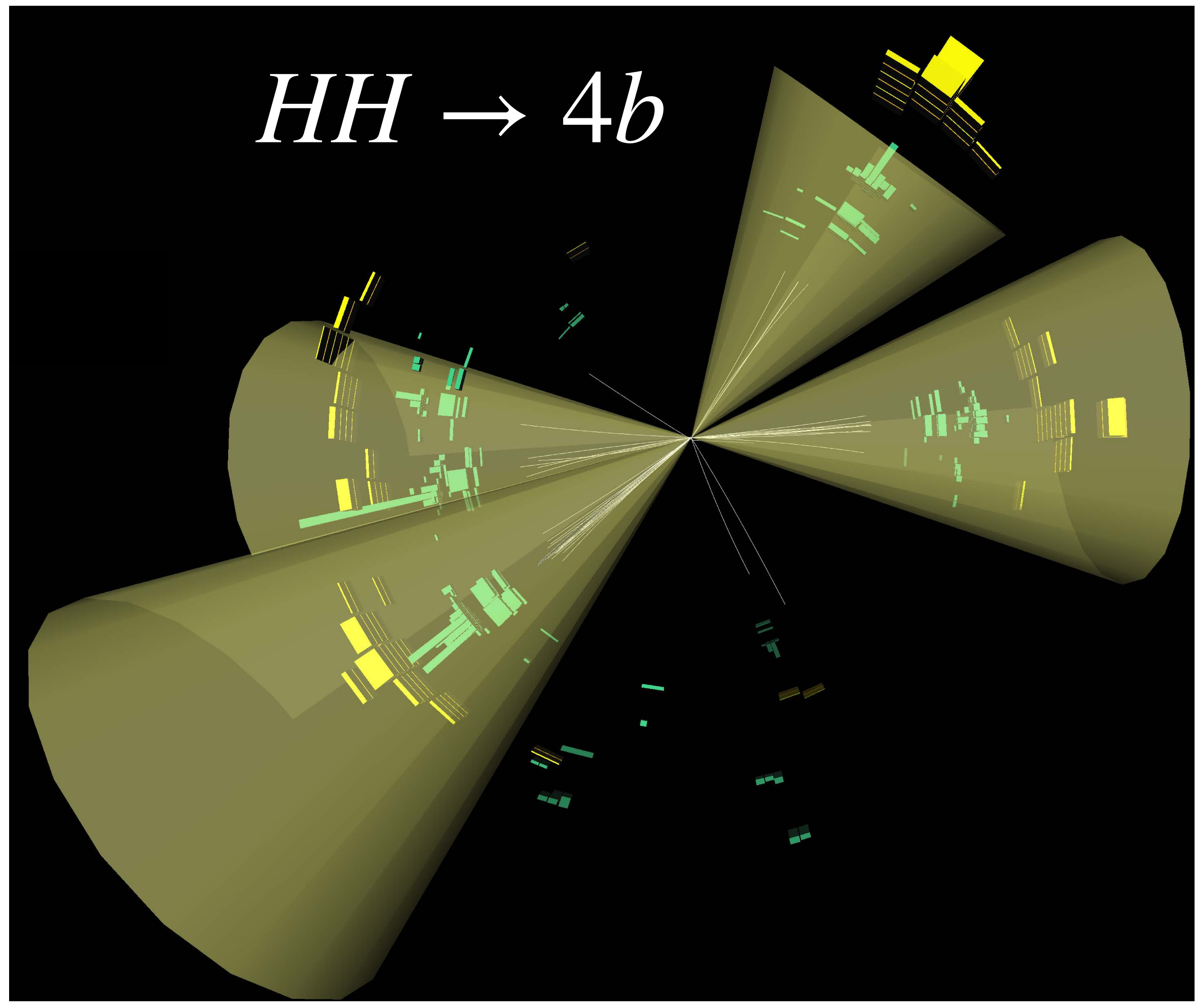
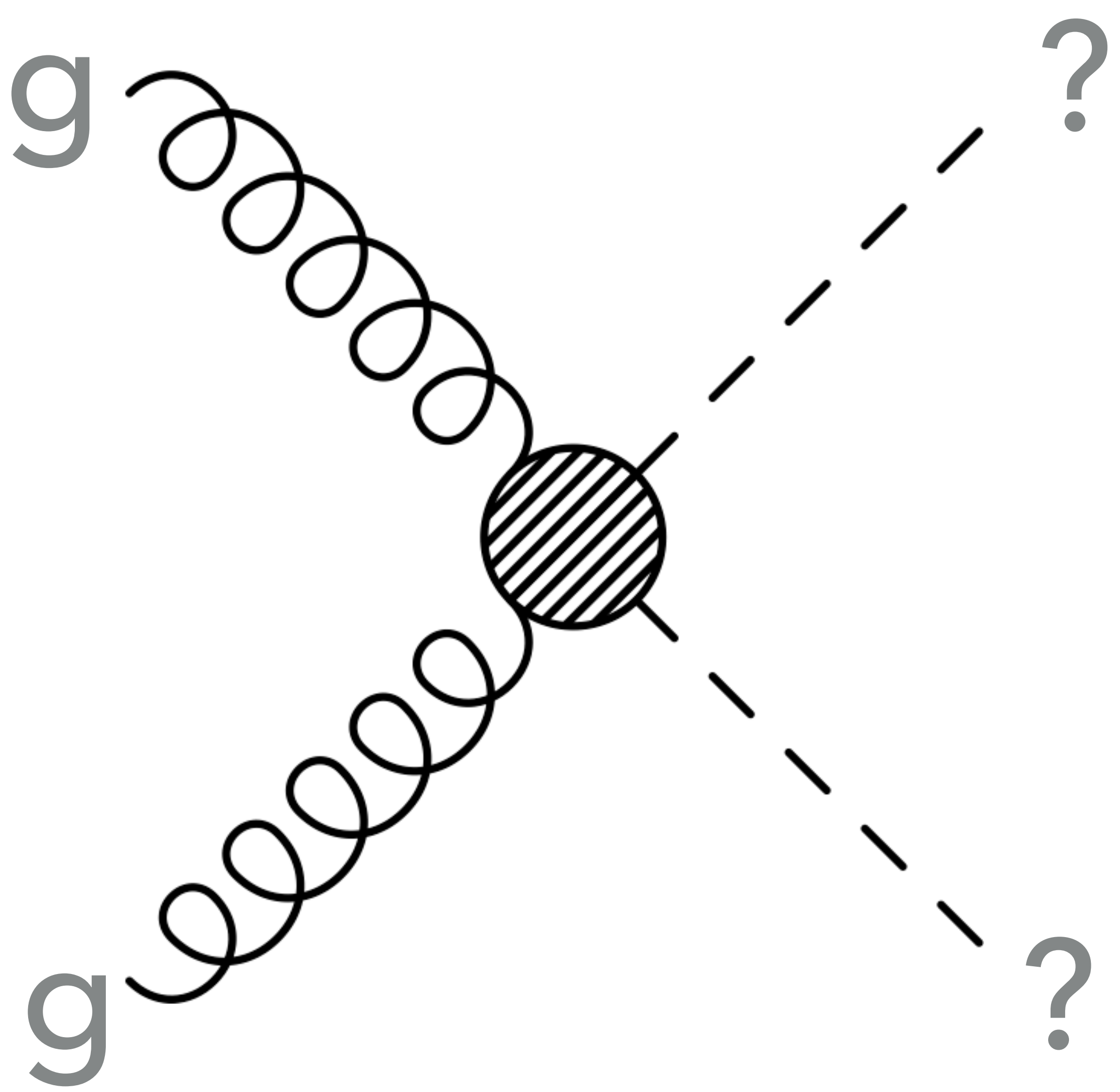


Image by Matt Strassler





WHAT COULD WE BE MISSING?

- How can we trigger on more complex low-energy hadronic signatures? Long-lived/displaced particles?
- What if we don't know exactly what to look for?
- What if our signatures require complex multivariate algorithms (e.g. b tagging)?
- How can we improve on our traditional (often slow) reconstruction algorithms?

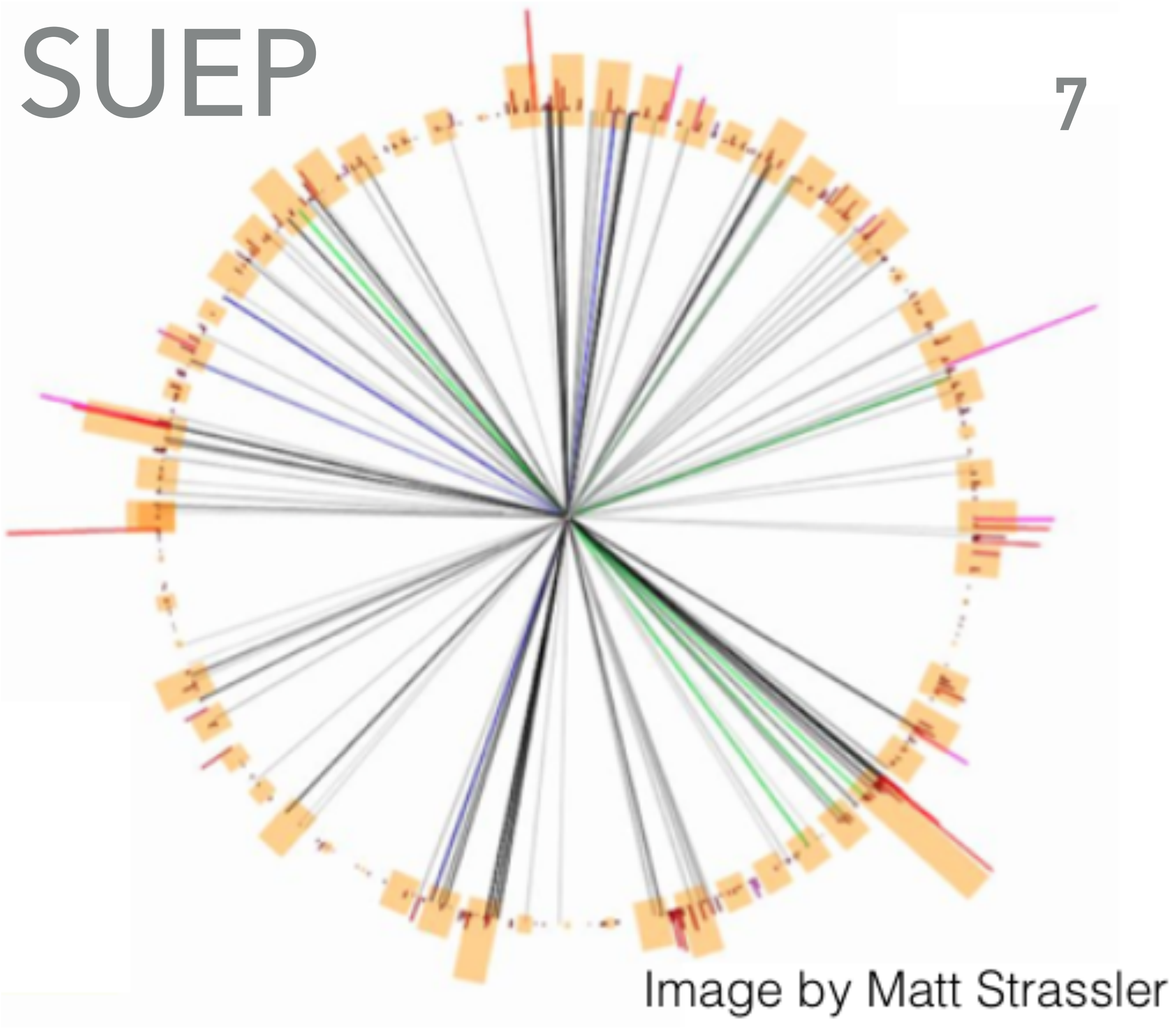
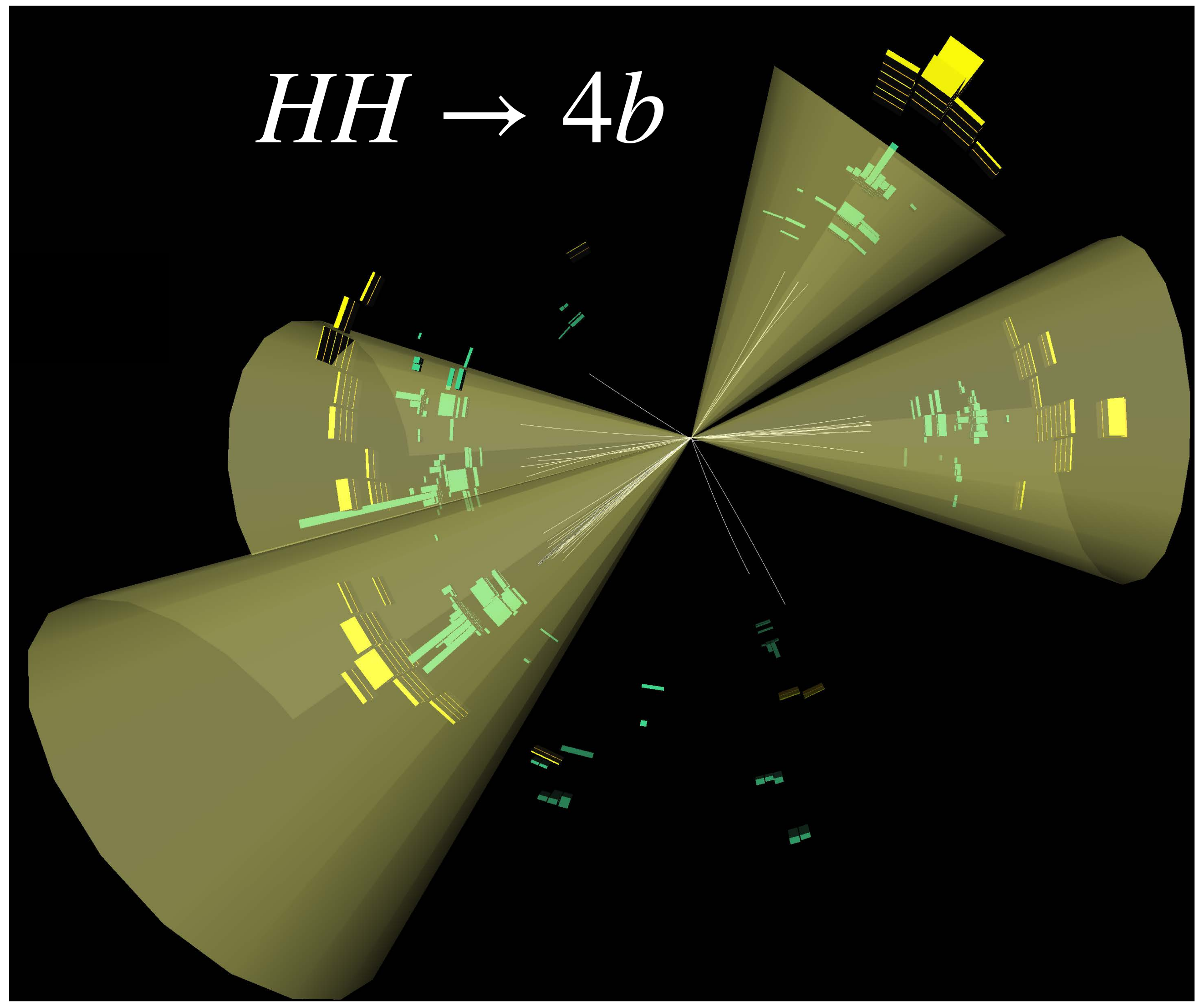
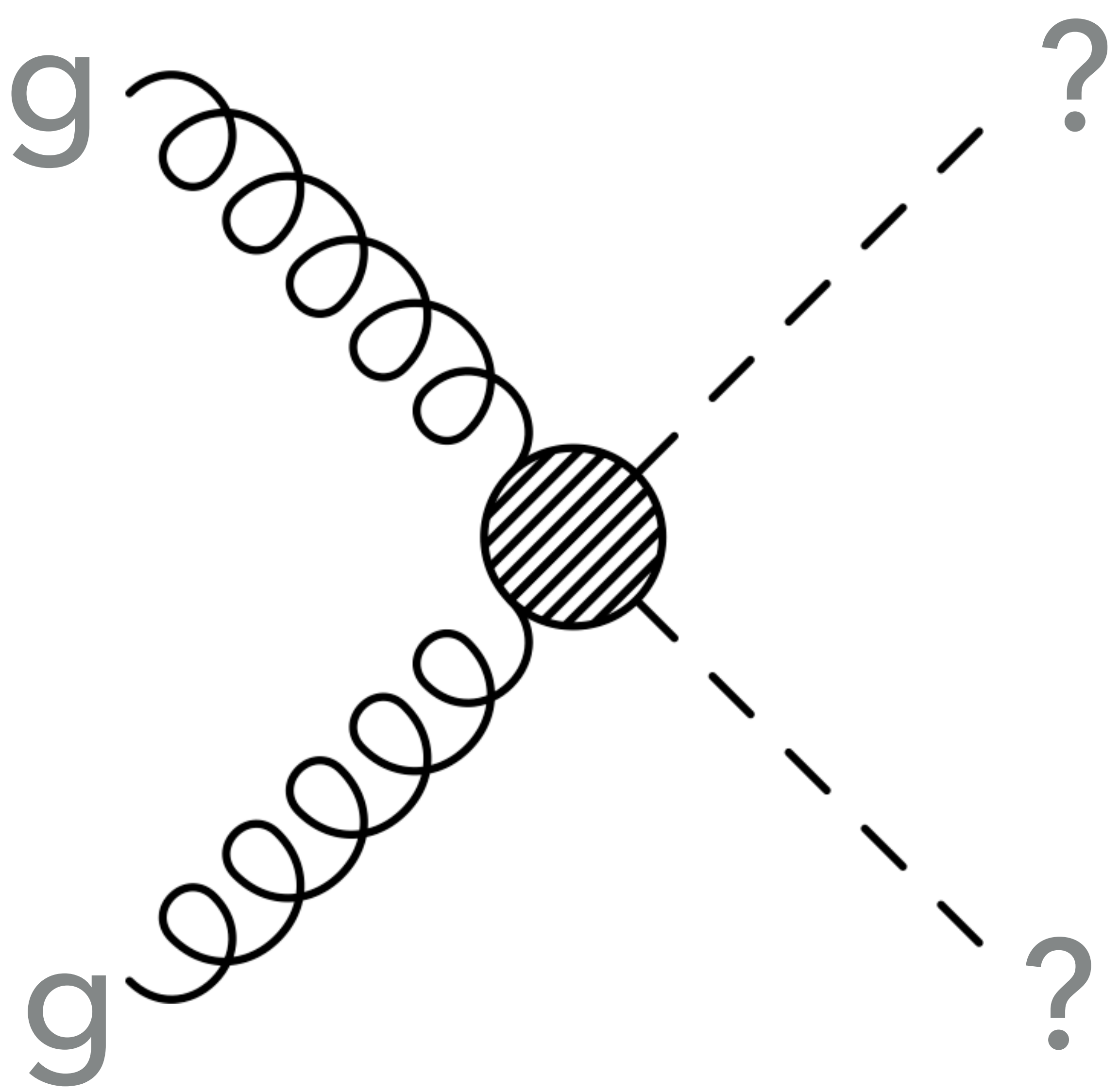
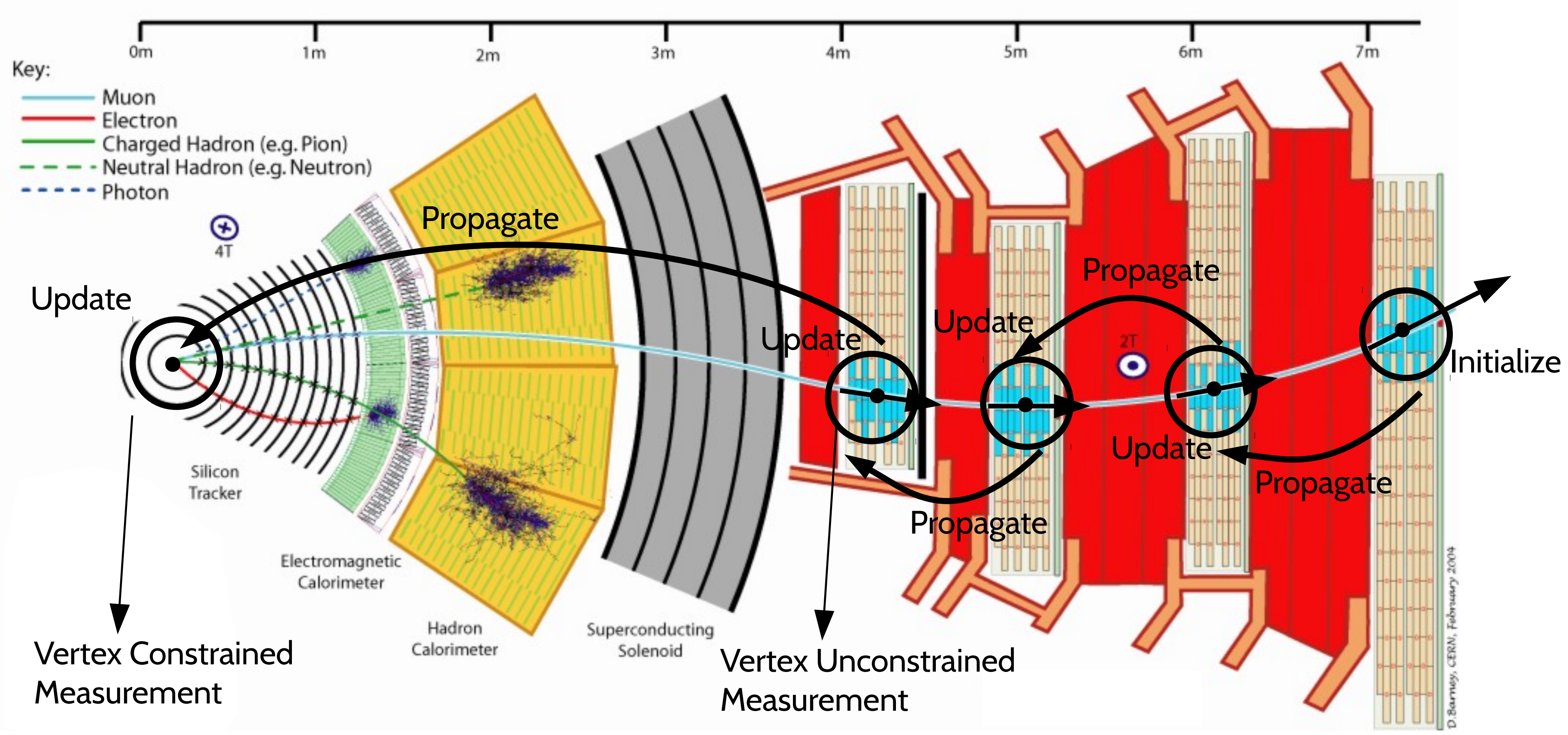


Image by Matt Strassler





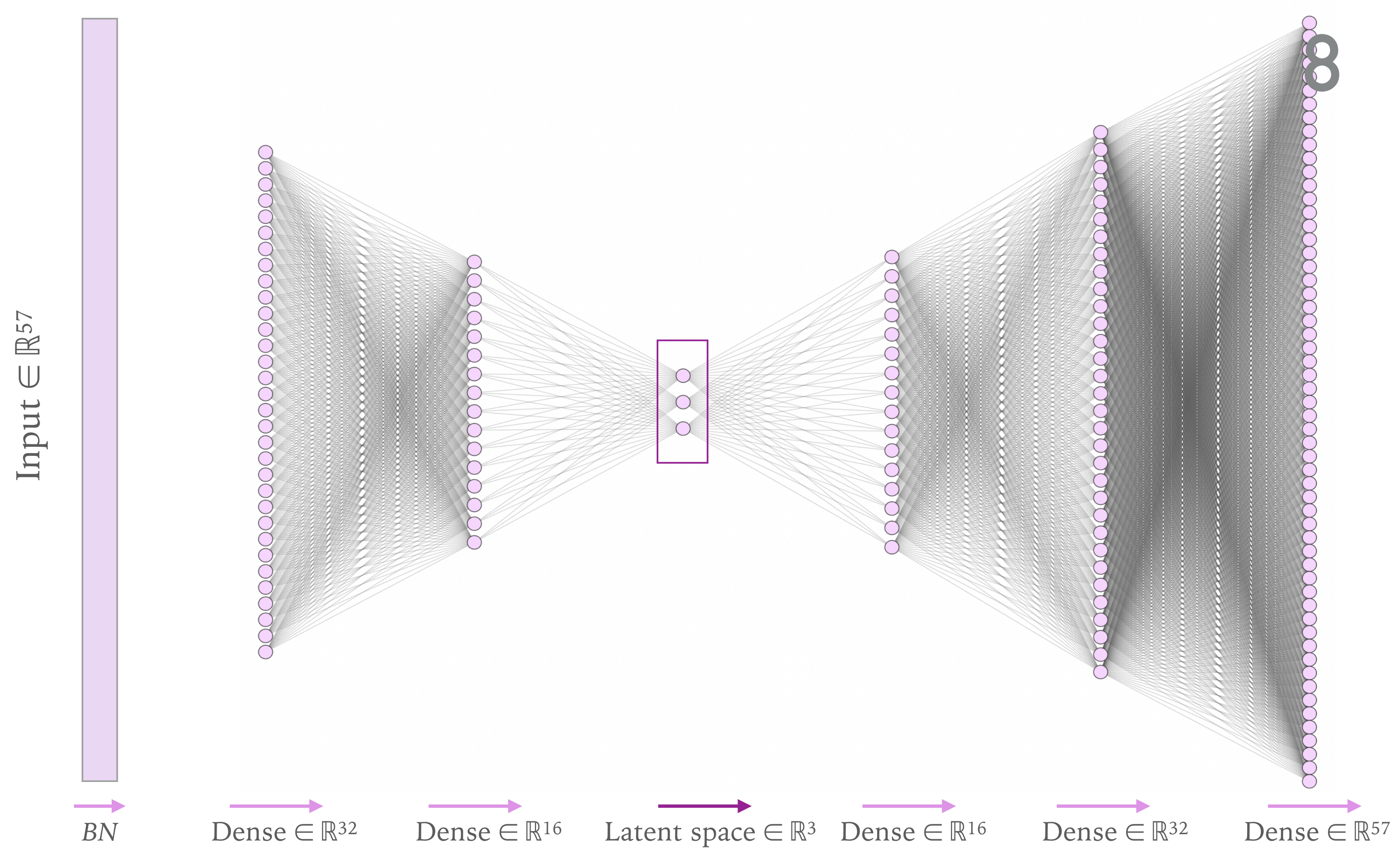


ML IN THE TRIGGER

ML IN THE TRIGGER

- (Variational) autoencoders for anomaly detection

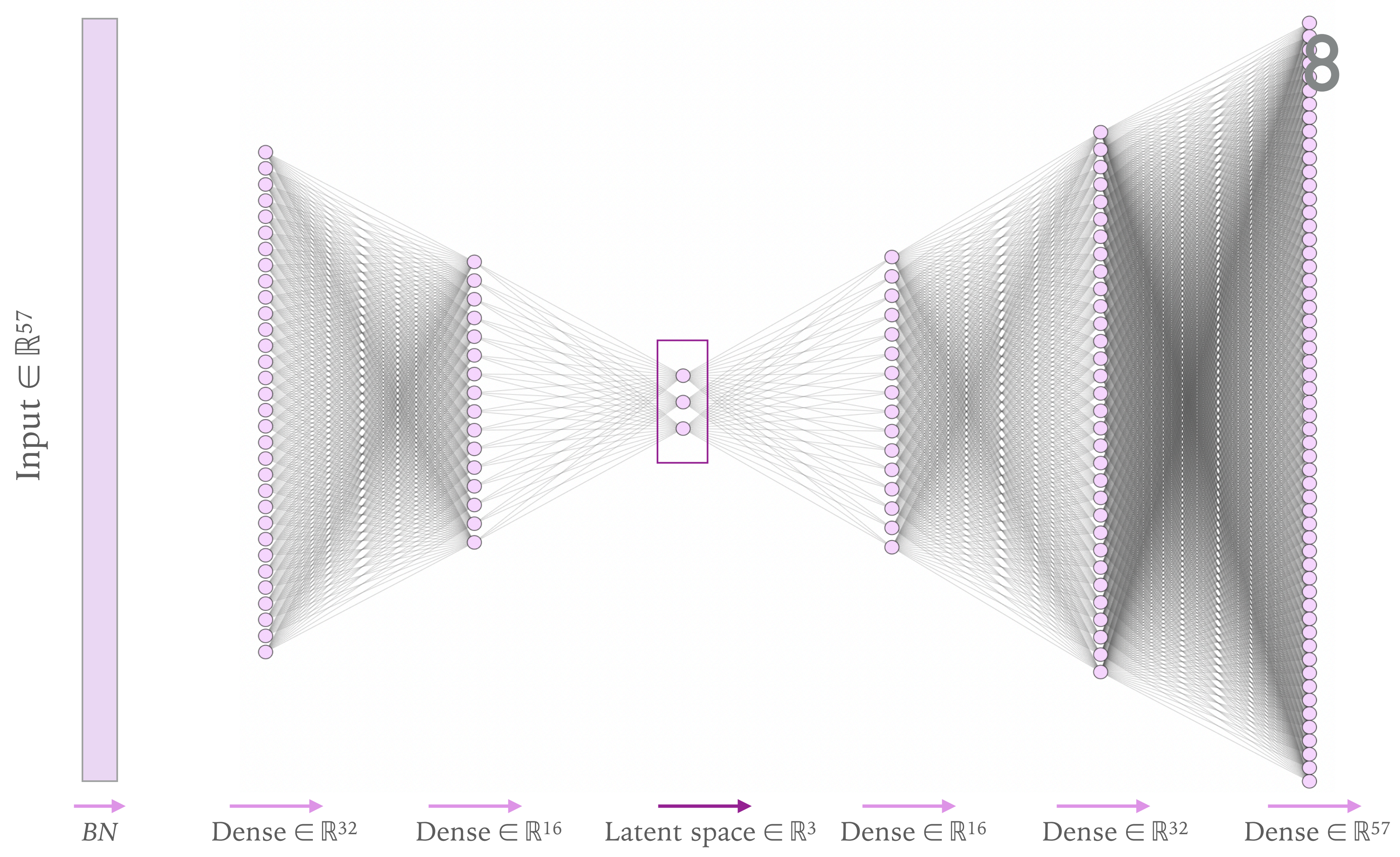
Nat. Mach. Intell. 4, 154 (2022)

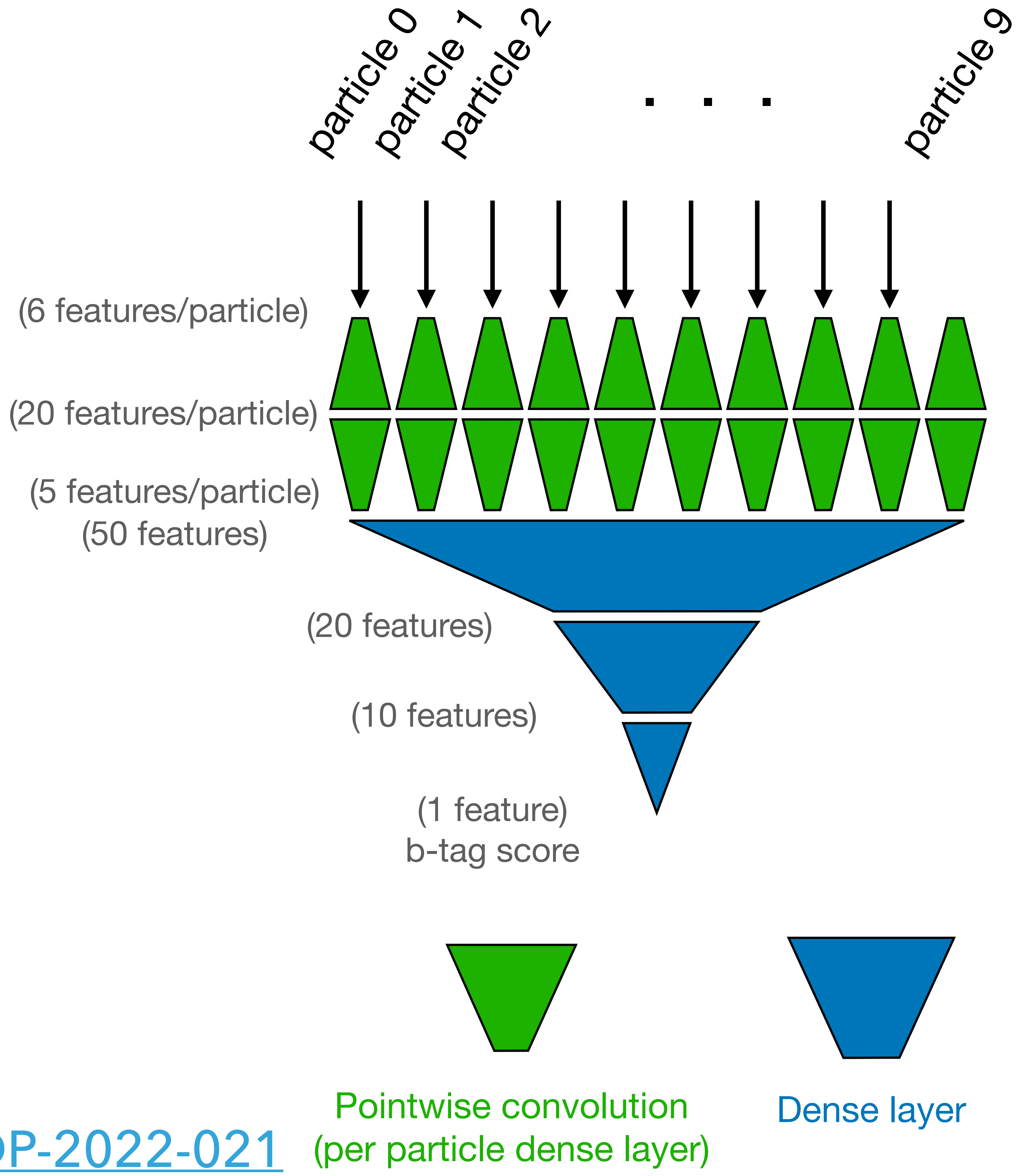


ML IN THE TRIGGER

Nat. Mach. Intell. 4, 154 (2022)

- (Variational) autoencoders for anomaly detection
- 1D convolutional neural networks for b-tagging



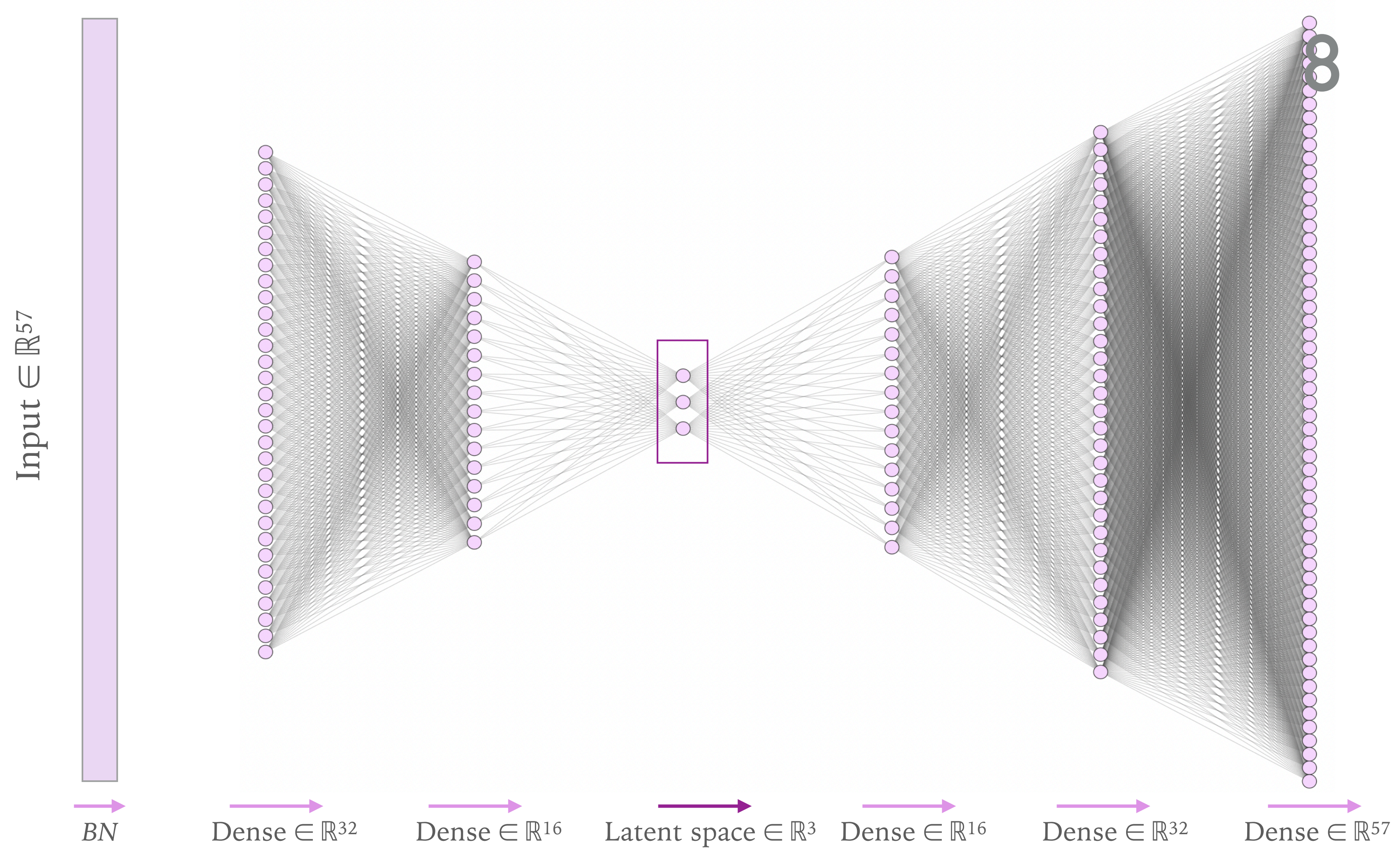


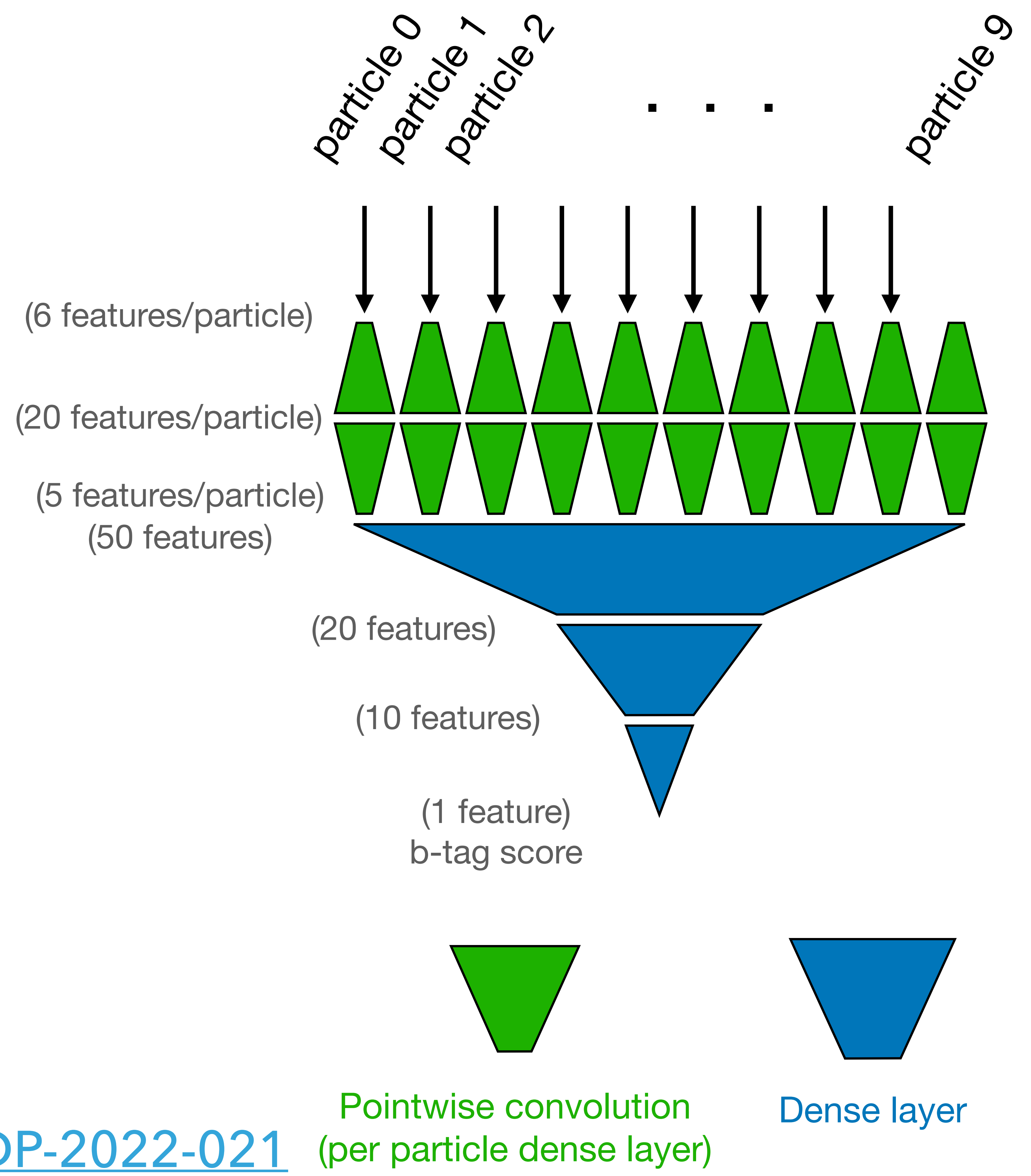
CMS-DP-2022-021

ML IN THE TRIGGER

- (Variational) autoencoders for anomaly detection
- 1D convolutional neural networks for b-tagging
- Graph neural networks for tracking

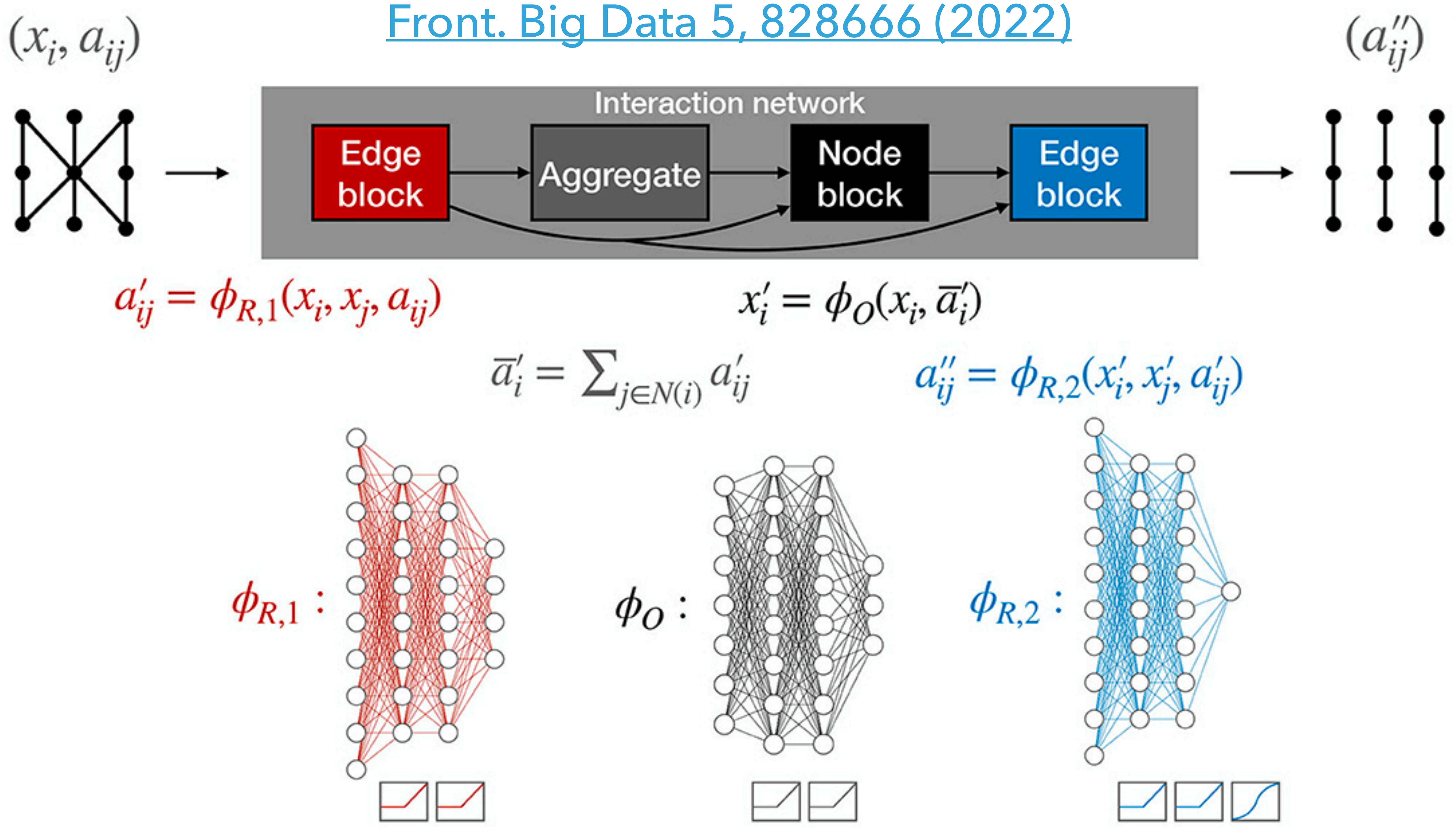
Nat. Mach. Intell. 4, 154 (2022)





CMS-DP-2022-021

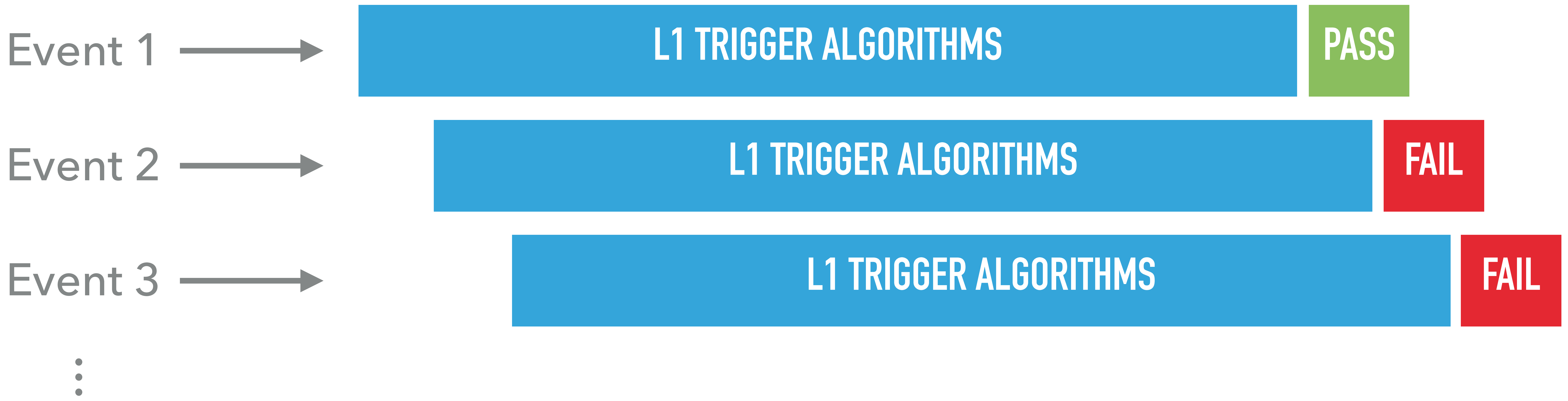
Front. Big Data 5, 828666 (2022)



WHAT MAKES THIS HARD?

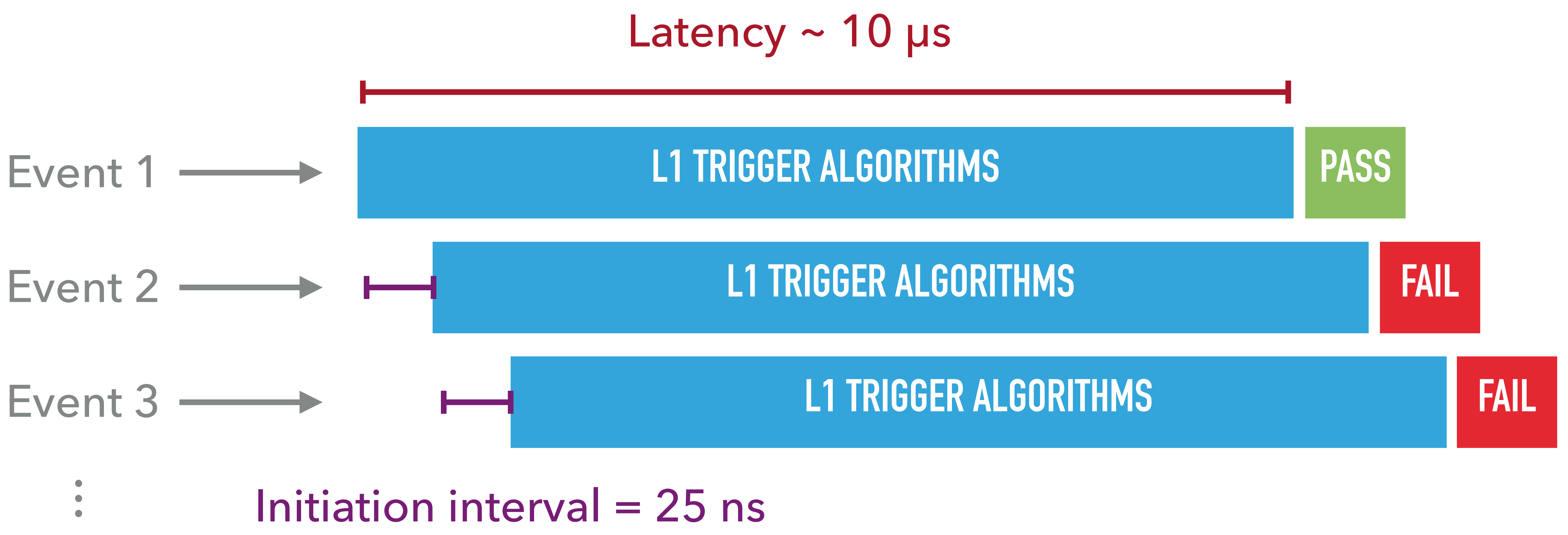
WHAT MAKES THIS HARD?

- Reconstruct all events and reject 98% of them in ~10 μs



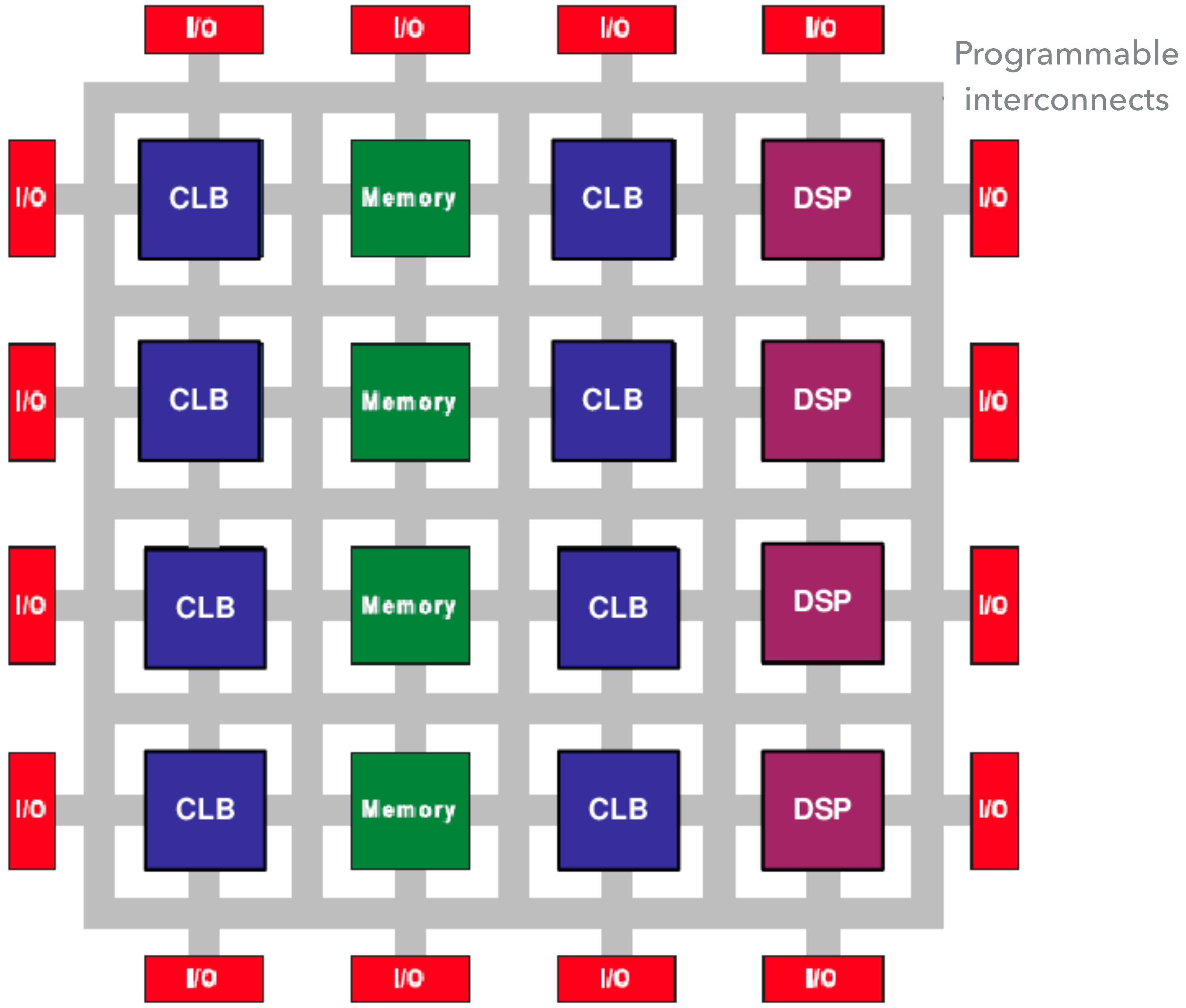
WHAT MAKES THIS HARD?

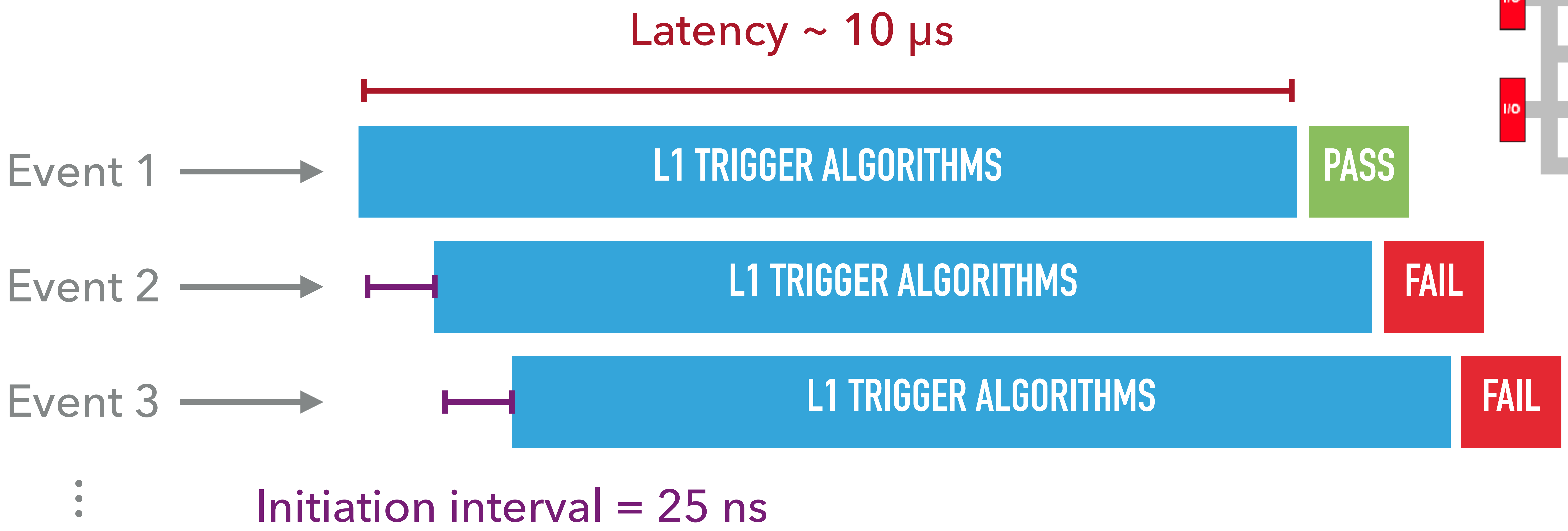
- Reconstruct all events and reject 98% of them in ~10 μs
  - Algorithms have to be <1 μs and process new events every (25 ns) × N_{tmux}



WHAT MAKES THIS HARD?

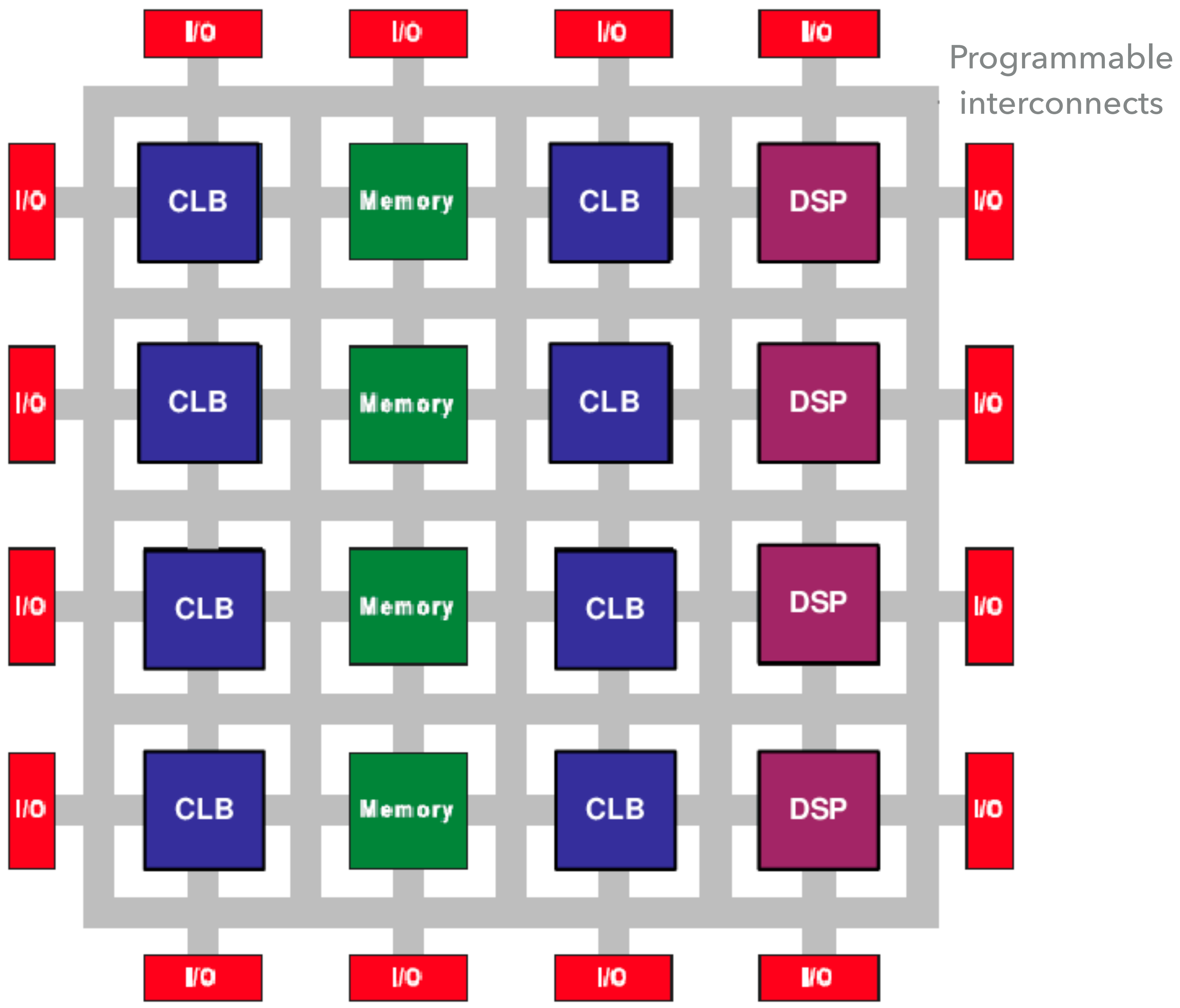
- Reconstruct all events and reject 98% of them in ~10 μs
  - Algorithms have to be <1 μs and process new events every (25 ns) × N_{tmux}
- Latency necessitates all **FPGA** design

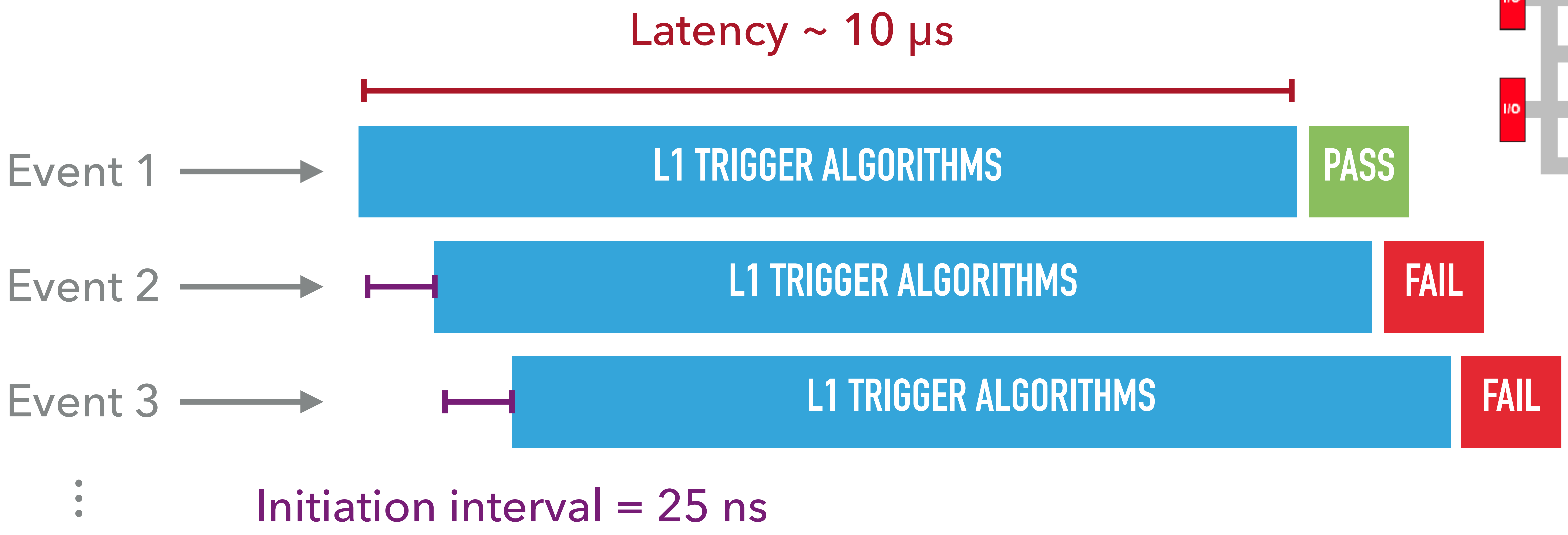




WHAT MAKES THIS HARD?

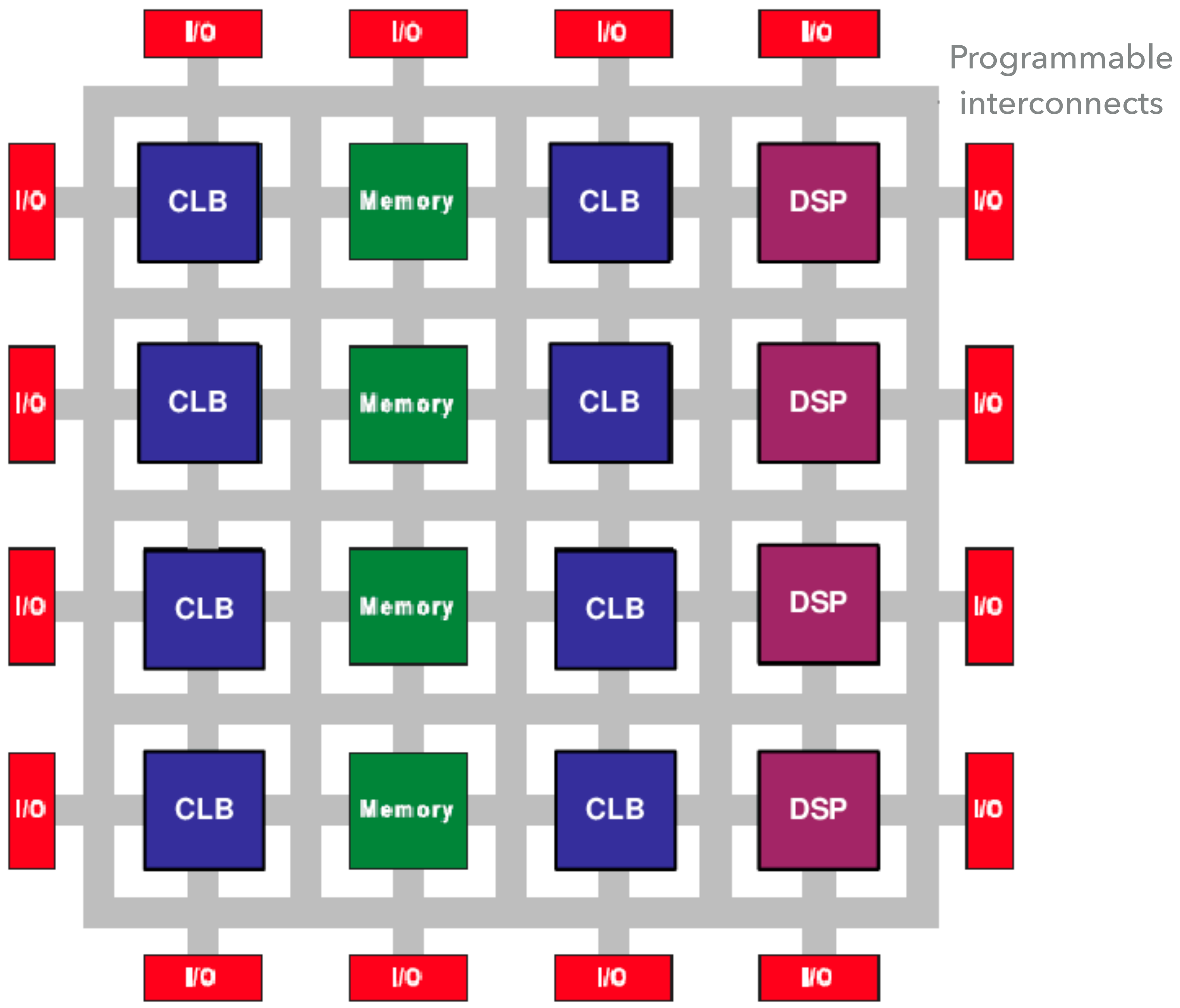
- Reconstruct all events and reject 98% of them in ~10 μs
  - Algorithms have to be <1 μs and process new events every (25 ns) × N_{tmux}
- Latency necessitates all **FPGA** design
  - Algorithms have to fit on <1 FPGA

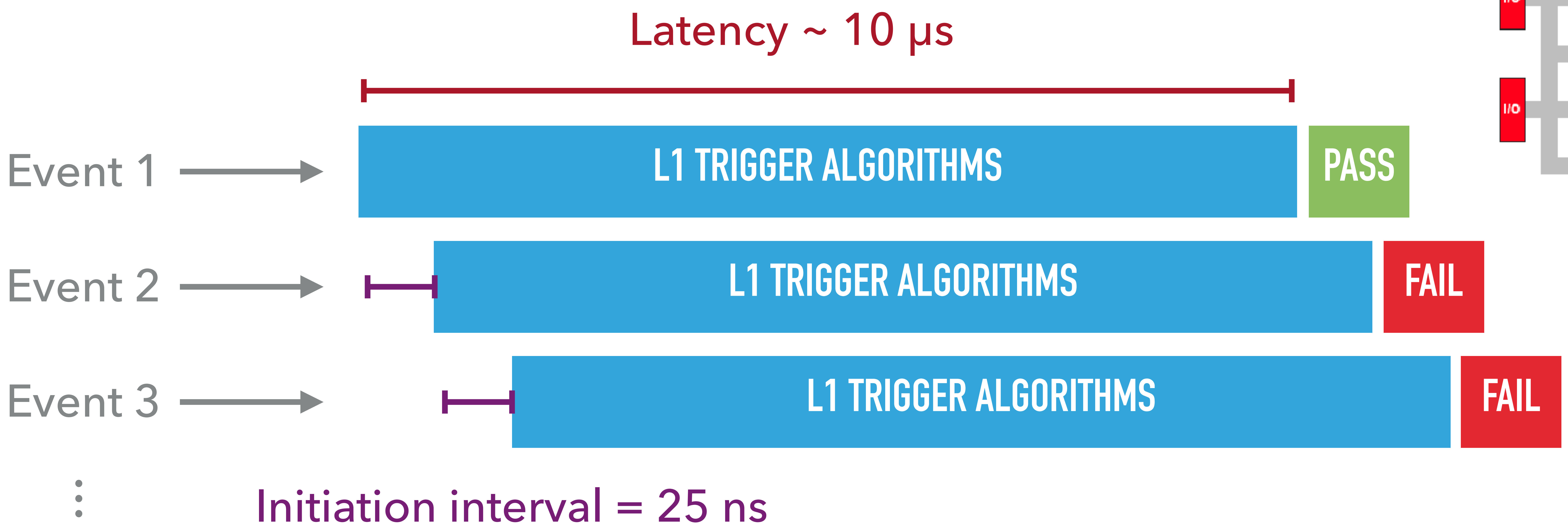




WHAT MAKES THIS HARD?

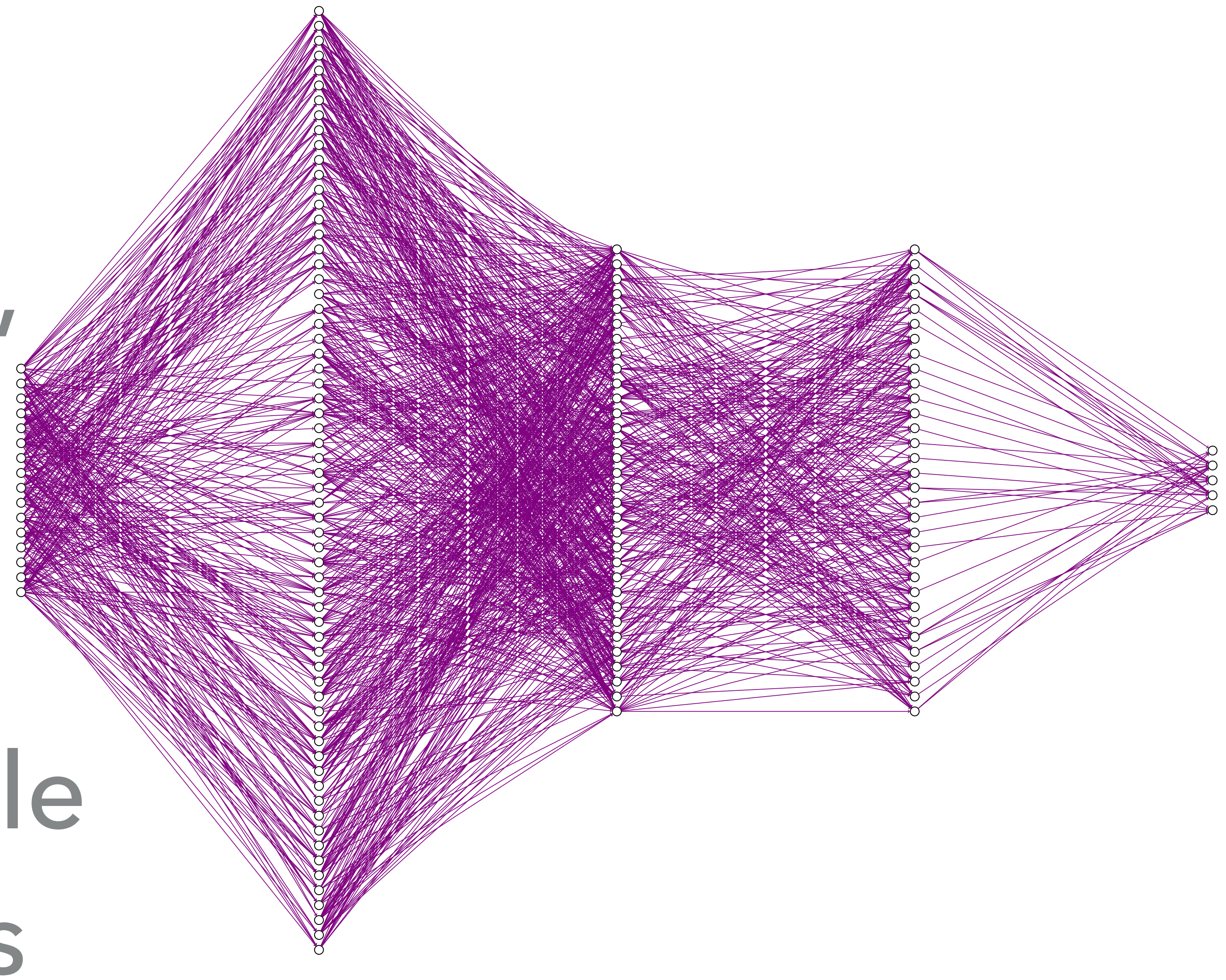
- Reconstruct all events and reject 98% of them in ~10 μs
  - Algorithms have to be <1 μs and process new events every (25 ns) × N_{tmux}
- Latency necessitates all **FPGA** design
  - Algorithms have to fit on <1 FPGA
- How can we satisfy these constraints?





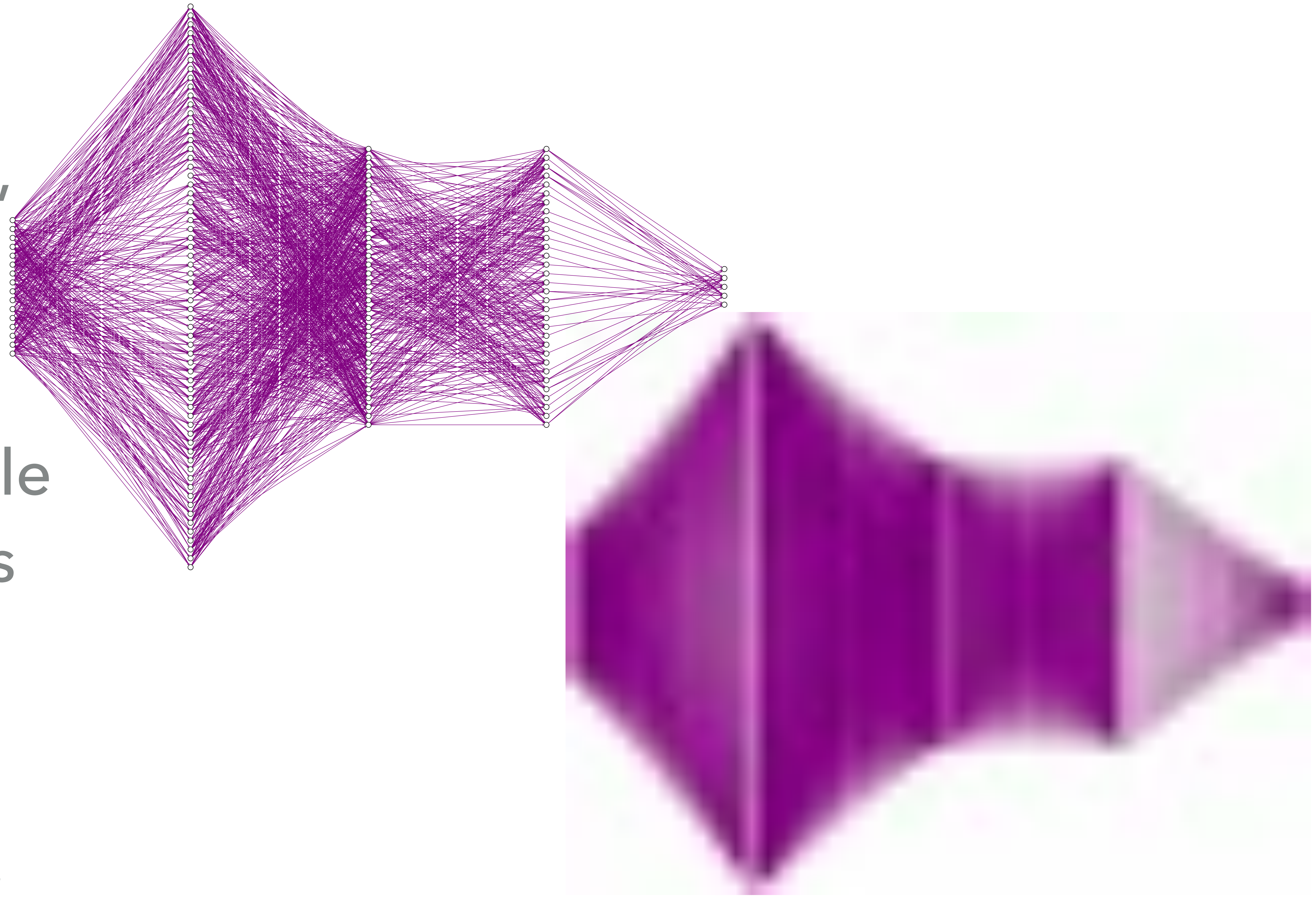
CODESIGN

- **Codesign**: intrinsic development loop between ML design, training, and implementation
- Pruning
  - Maintain high performance while removing redundant operations



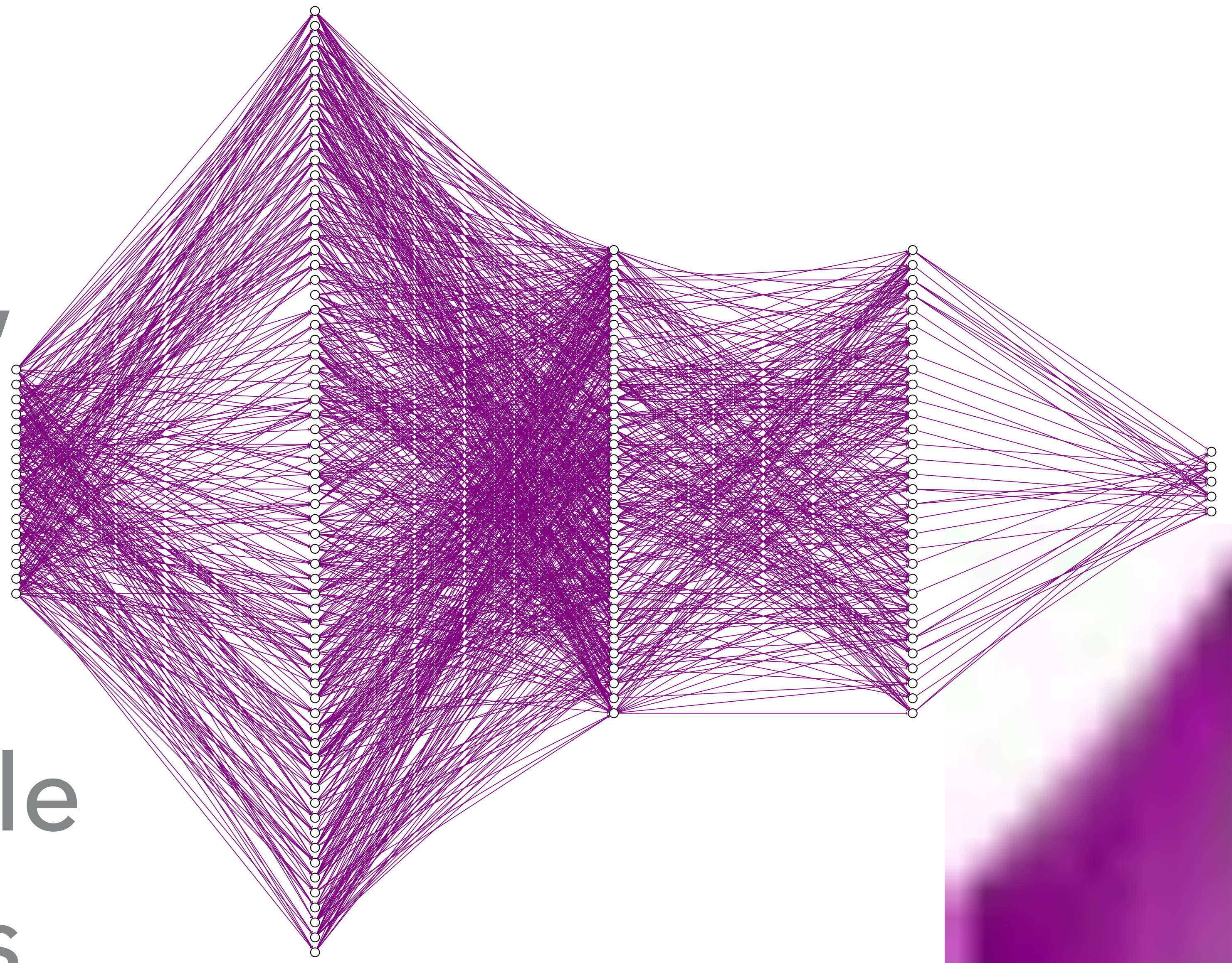
CODESIGN

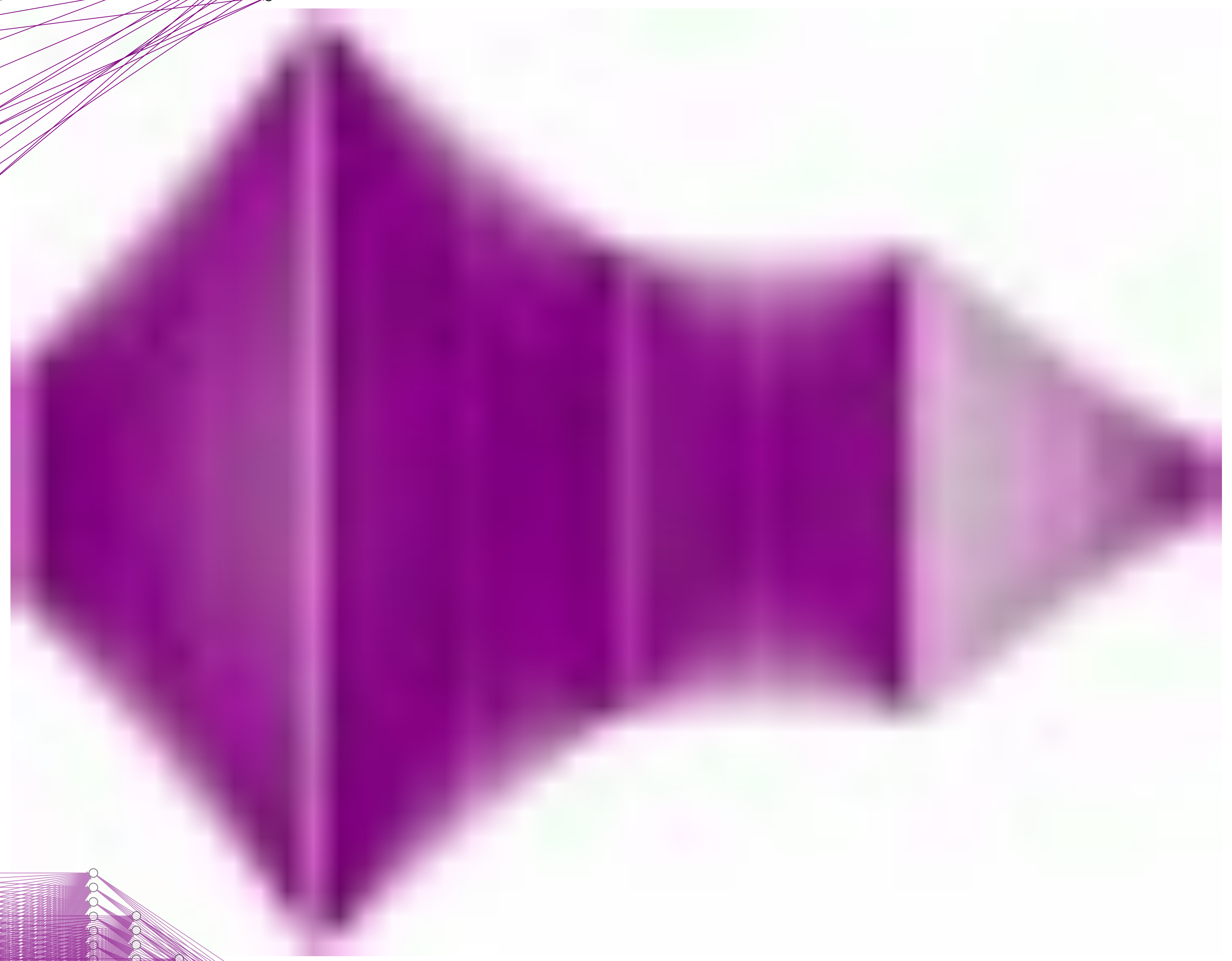
- **Codesign**: intrinsic development loop between ML design, training, and implementation
- Pruning
  - Maintain high performance while removing redundant operations
- Quantization
  - Reduce precision from 32-bit floating point to 16-bit, 8-bit, …

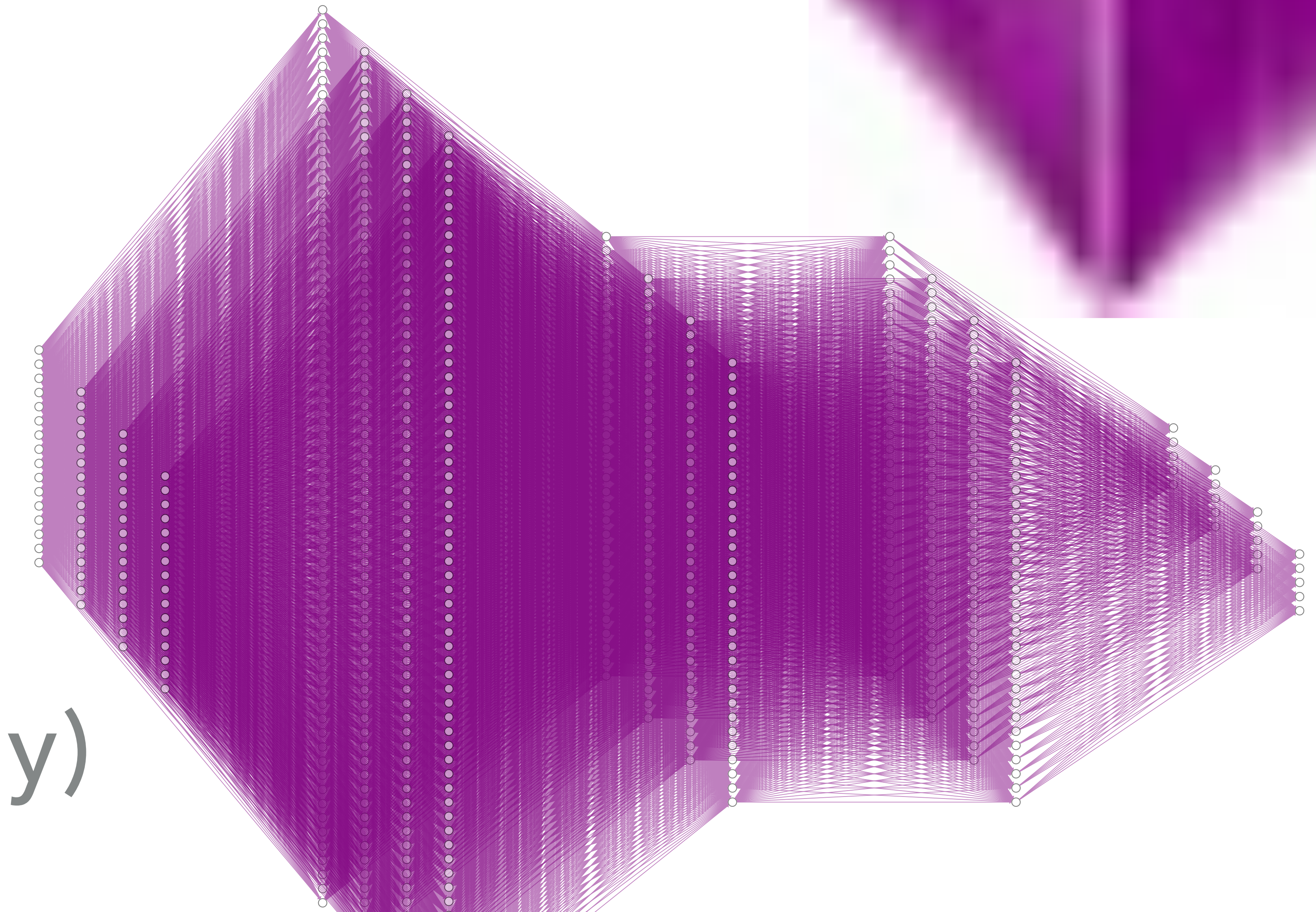


CODESIGN

- **Codesign**: intrinsic development loop between ML design, training, and implementation
- Pruning
  - Maintain high performance while removing redundant operations
- Quantization
  - Reduce precision from 32-bit floating point to 16-bit, 8-bit, …
- Parallelization
  - Balance parallelization (how fast) with resources needed (how costly)







I. INTRO & MOTIVATION

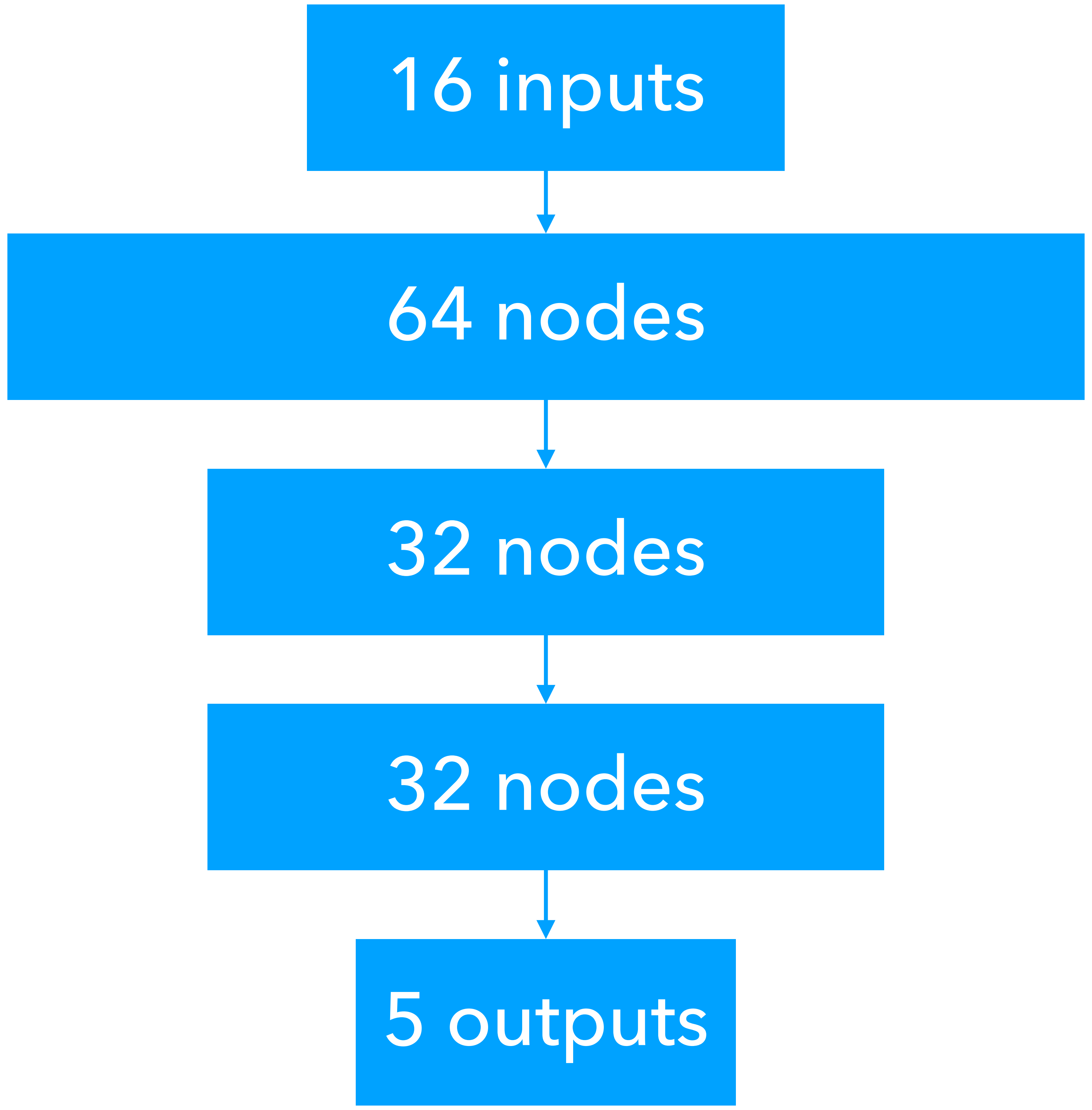
II. COMPRESSION

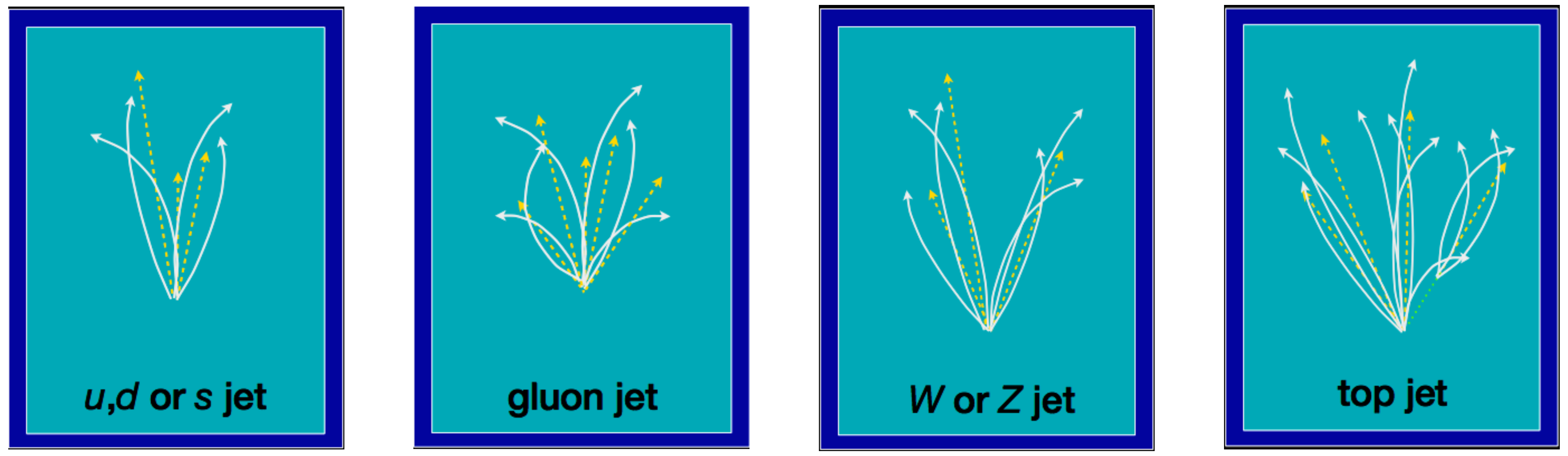
III. HARDWARE

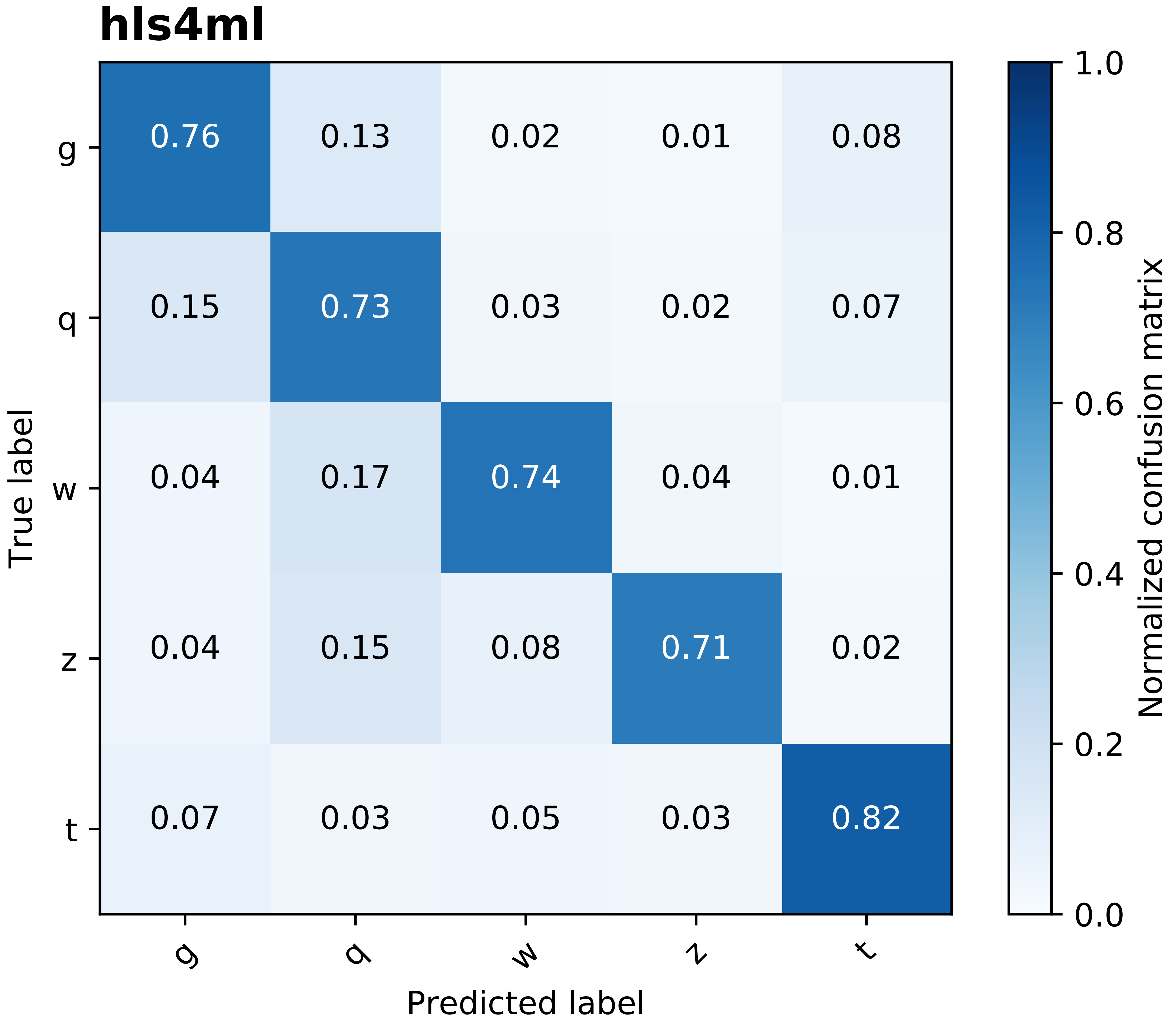
IV. APPLICATIONS

BENCHMARK: JET TAGGING MLP

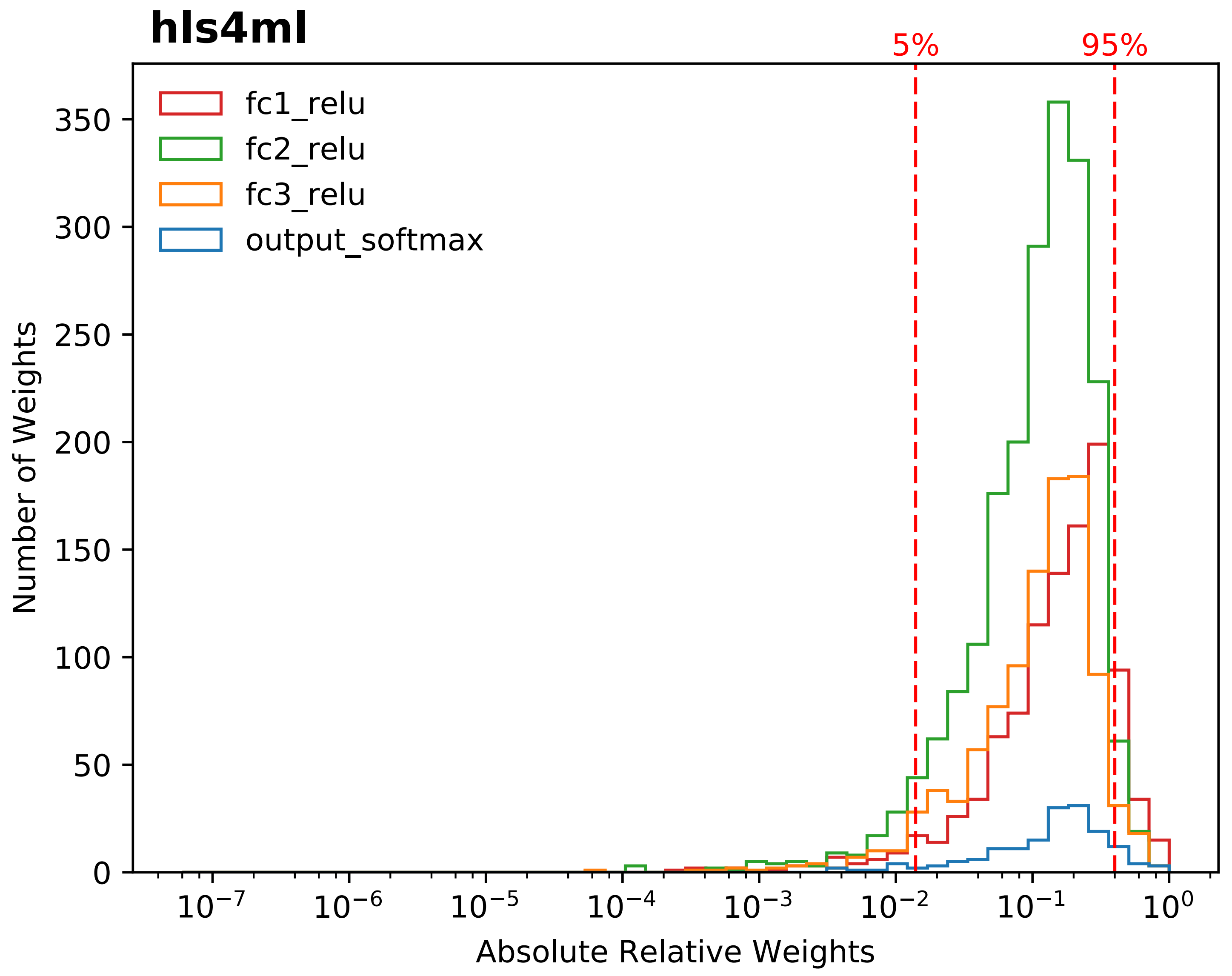
Small NN benchmark correctly identifies particle "jets" 70-80% of the time







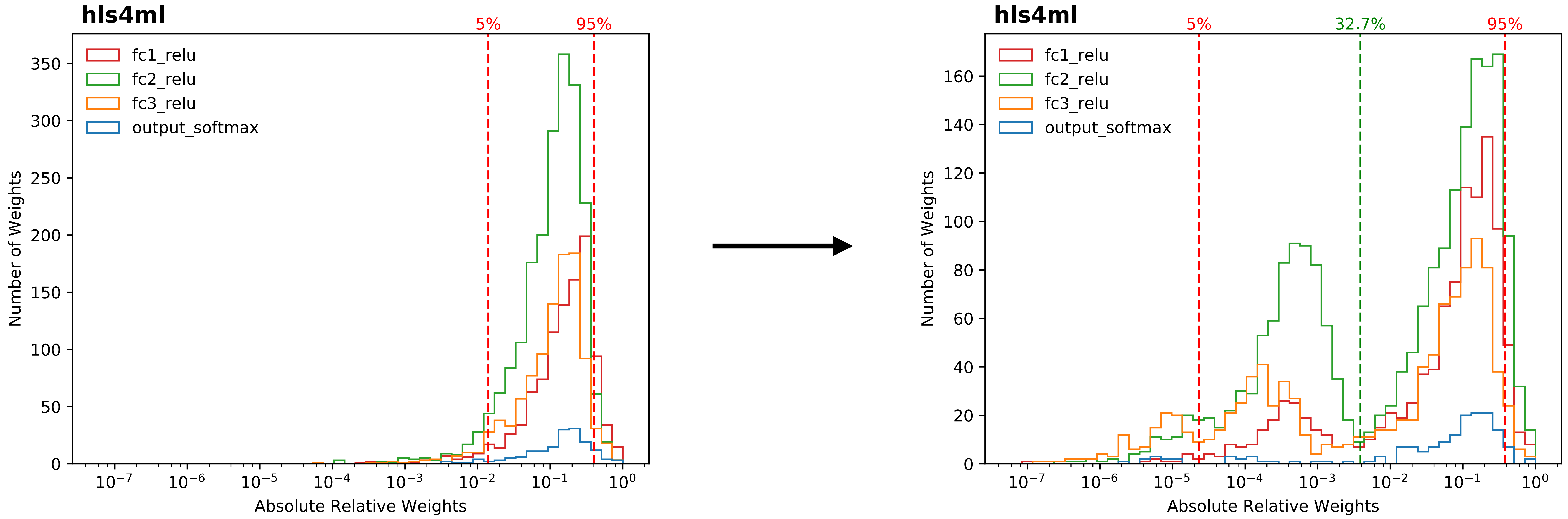
ITERATIVE MAGNITUDE-BASED PRUNING



ITERATIVE MAGNITUDE-BASED PRUNING

- Train with **L_1 regularization** (down-weights unimportant synapses)

$$L_\lambda(w) = L(w) + \lambda \|w\|_1 \qquad \|w\|_1 = \sum_i |w_i|$$

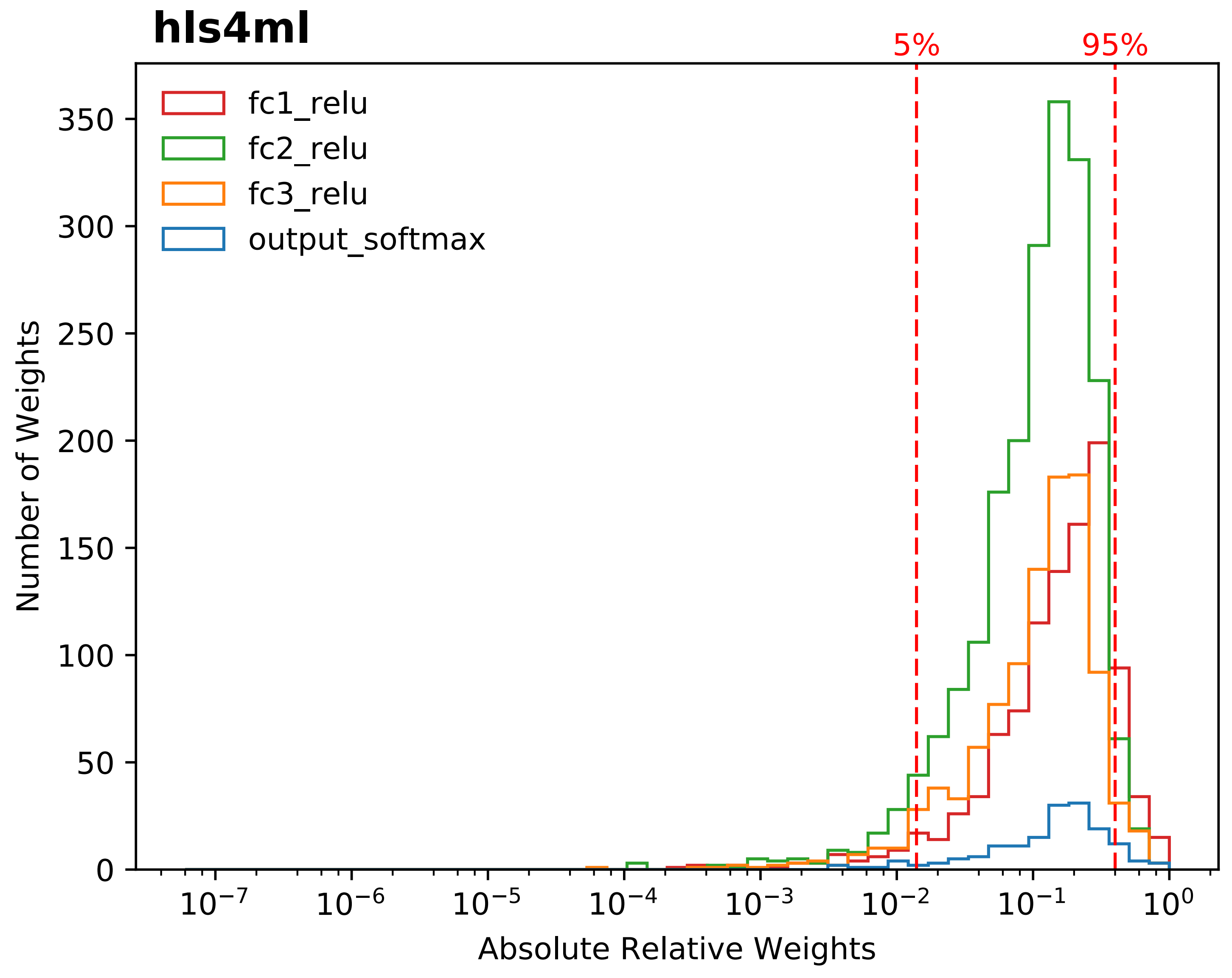


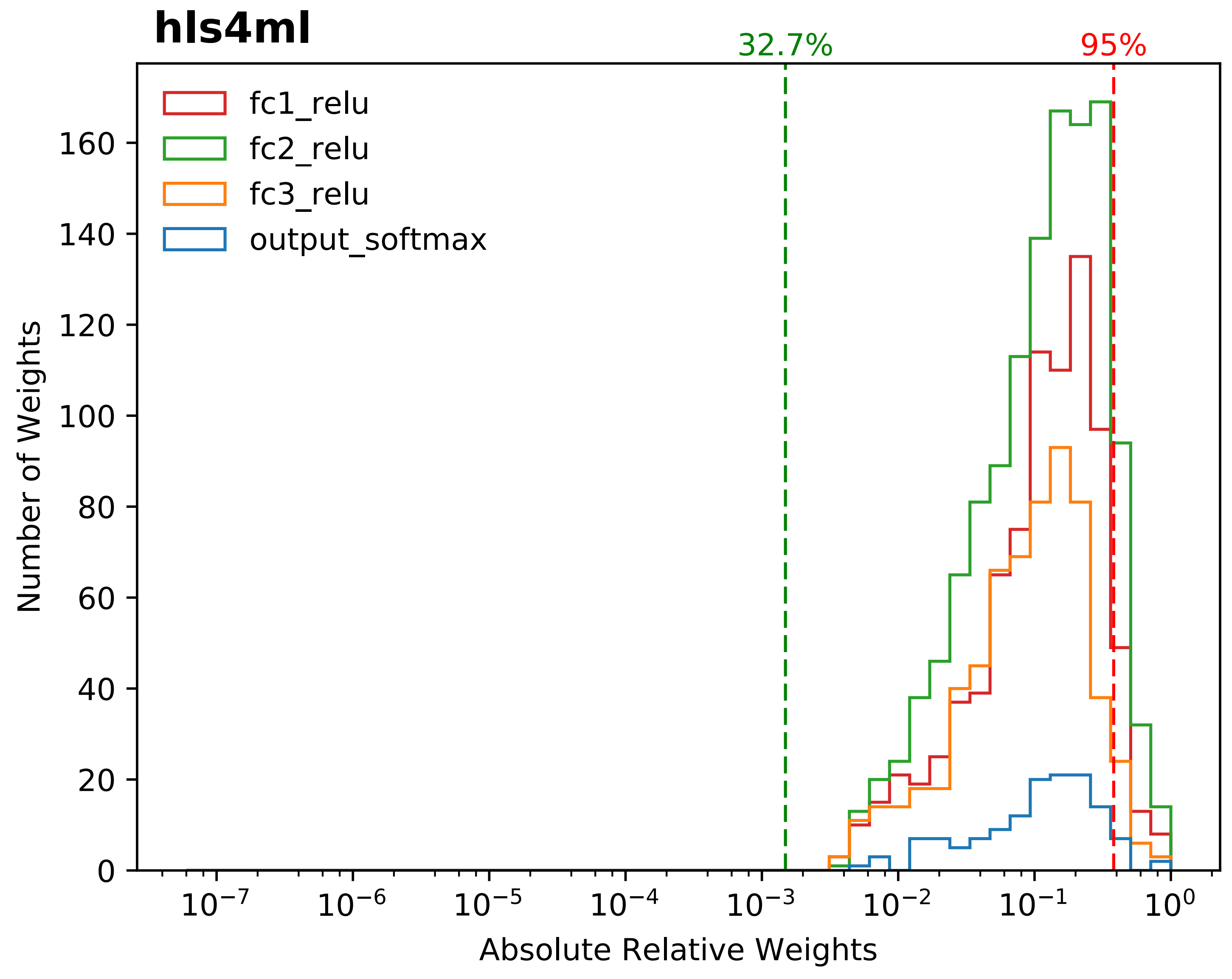
ITERATIVE MAGNITUDE-BASED PRUNING

- Train with **L_1 regularization** (down-weights unimportant synapses)

$$L_\lambda(\boldsymbol{w}) = L(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_1 \qquad \|\boldsymbol{w}\|_1 = \sum_i |w_i|$$

- Remove **smallest** weights



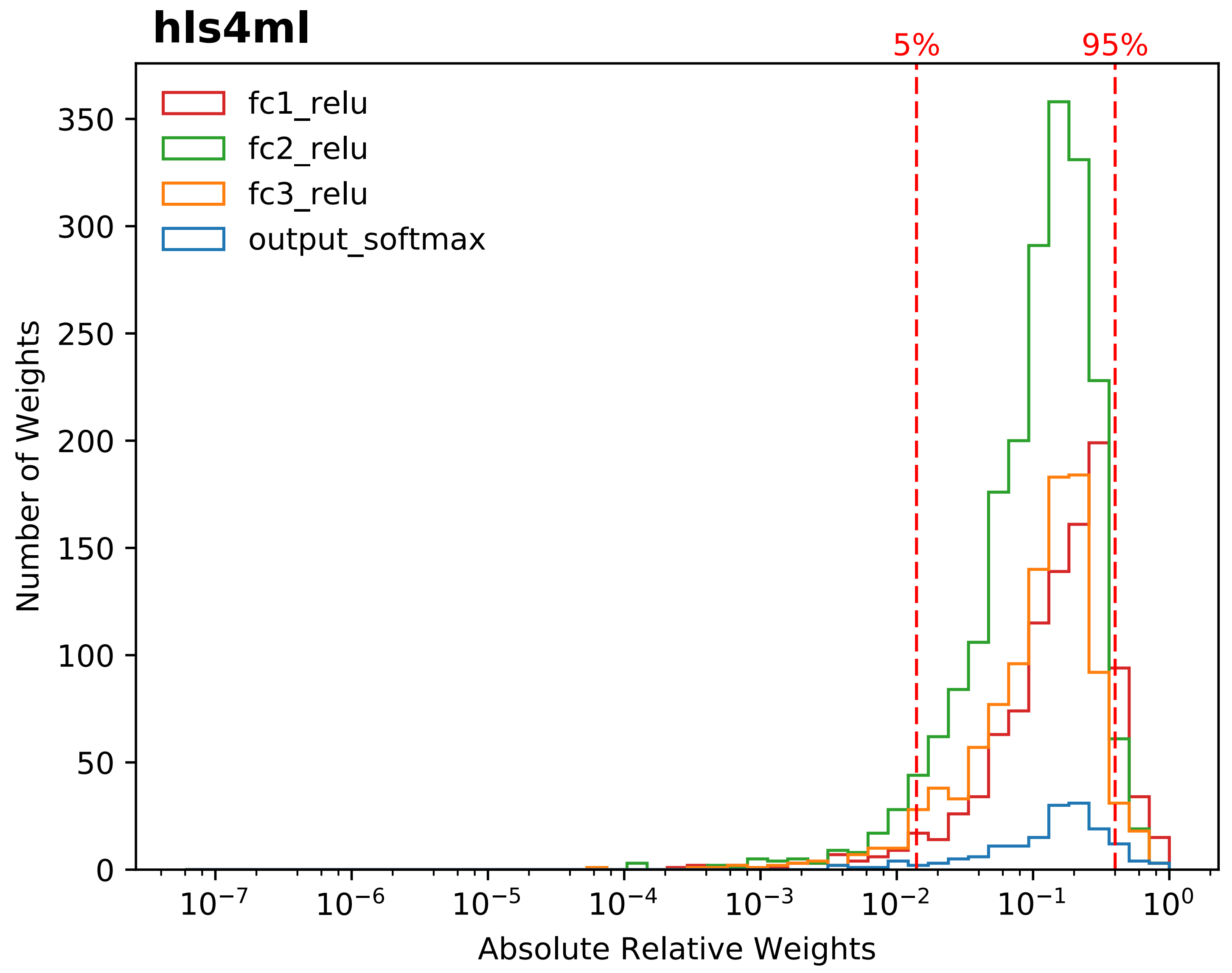


- Train with **L_1 regularization** (down-weights unimportant synapses)

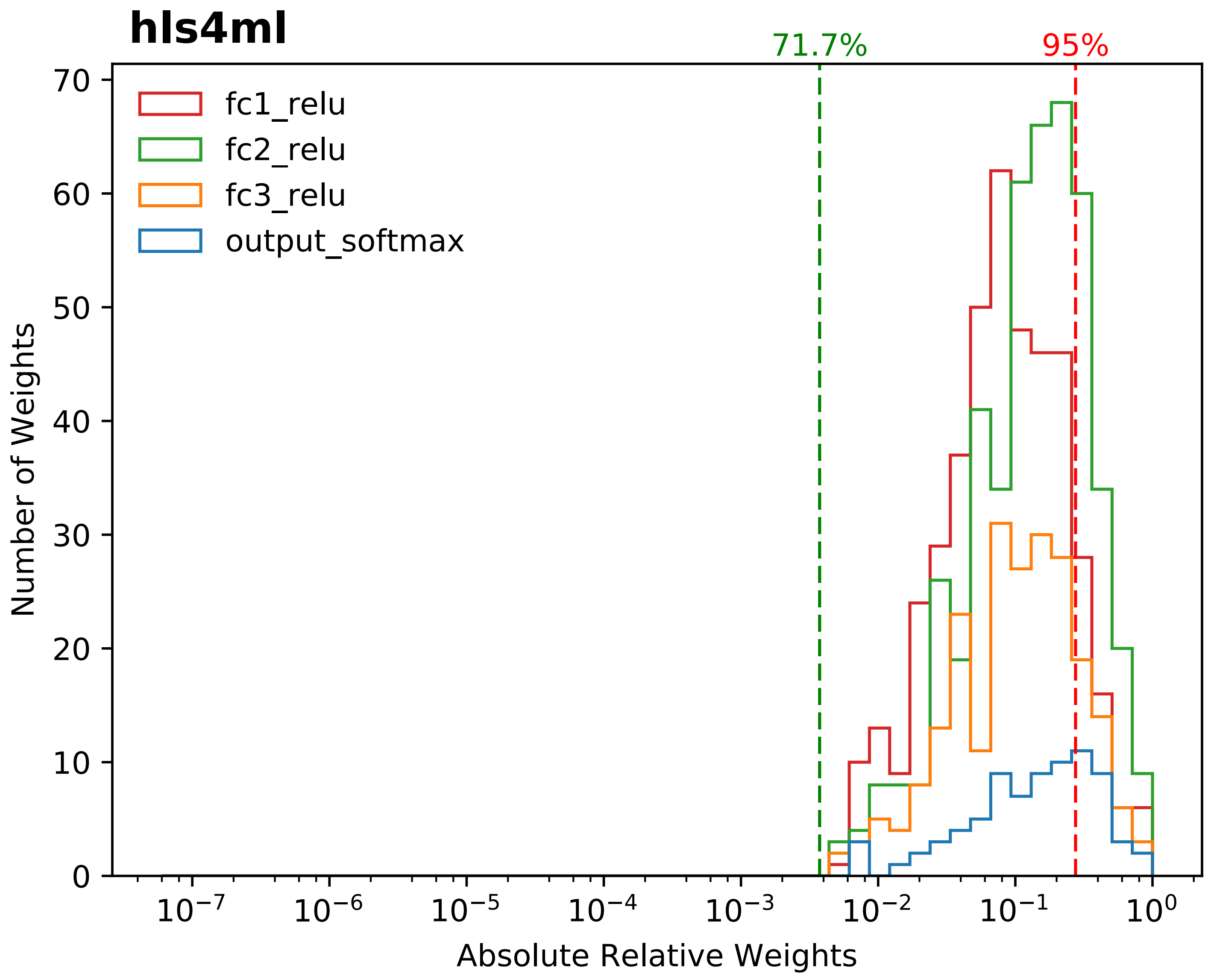
$$L_\lambda(\boldsymbol{w}) = L(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_1 \qquad \|\boldsymbol{w}\|_1 = \sum_i |w_i|$$

- Remove **smallest** weights
- Iterate

70% REDUCTION OF WEIGHTS WITH NO LOSS IN PERF.



→



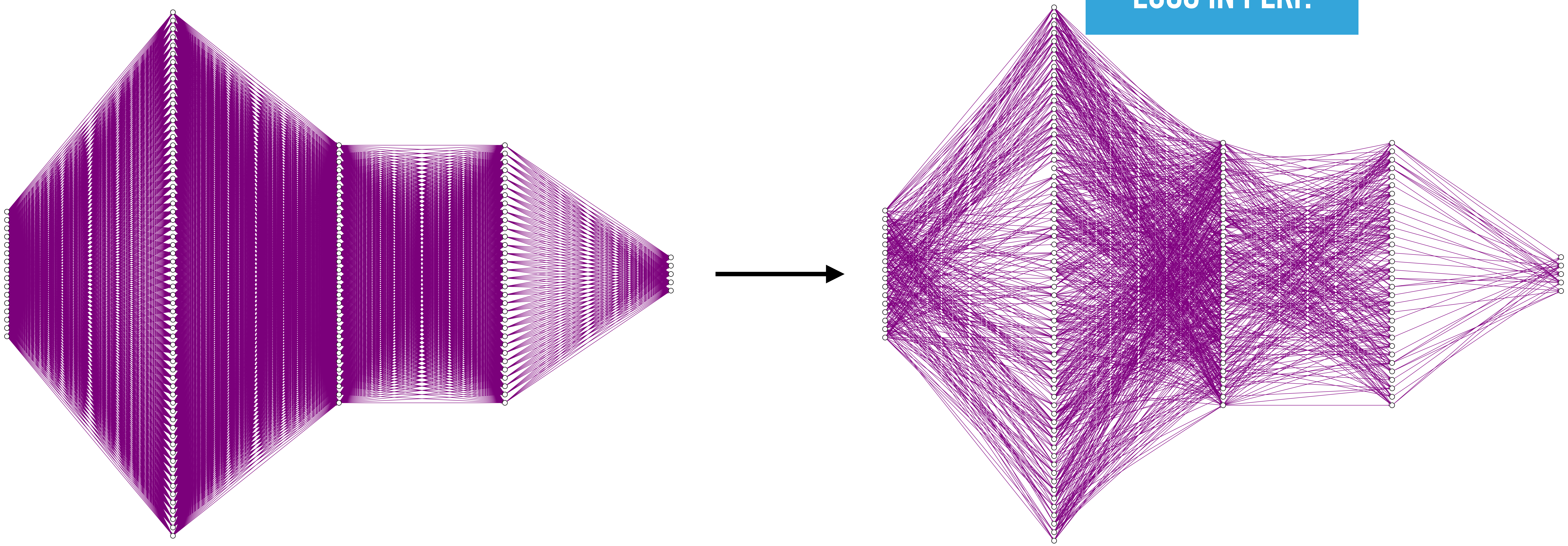
ITERATIVE MAGNITUDE-BASED PRUNING

- Train with **L_1 regularization** (down-weights unimportant synapses)

$$L_\lambda(\boldsymbol{w}) = L(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_1 \qquad \|\boldsymbol{w}\|_1 = \sum_i |w_i|$$

- Remove **smallest** weights
- Iterate

70% REDUCTION OF WEIGHTS WITH NO LOSS IN PERF.



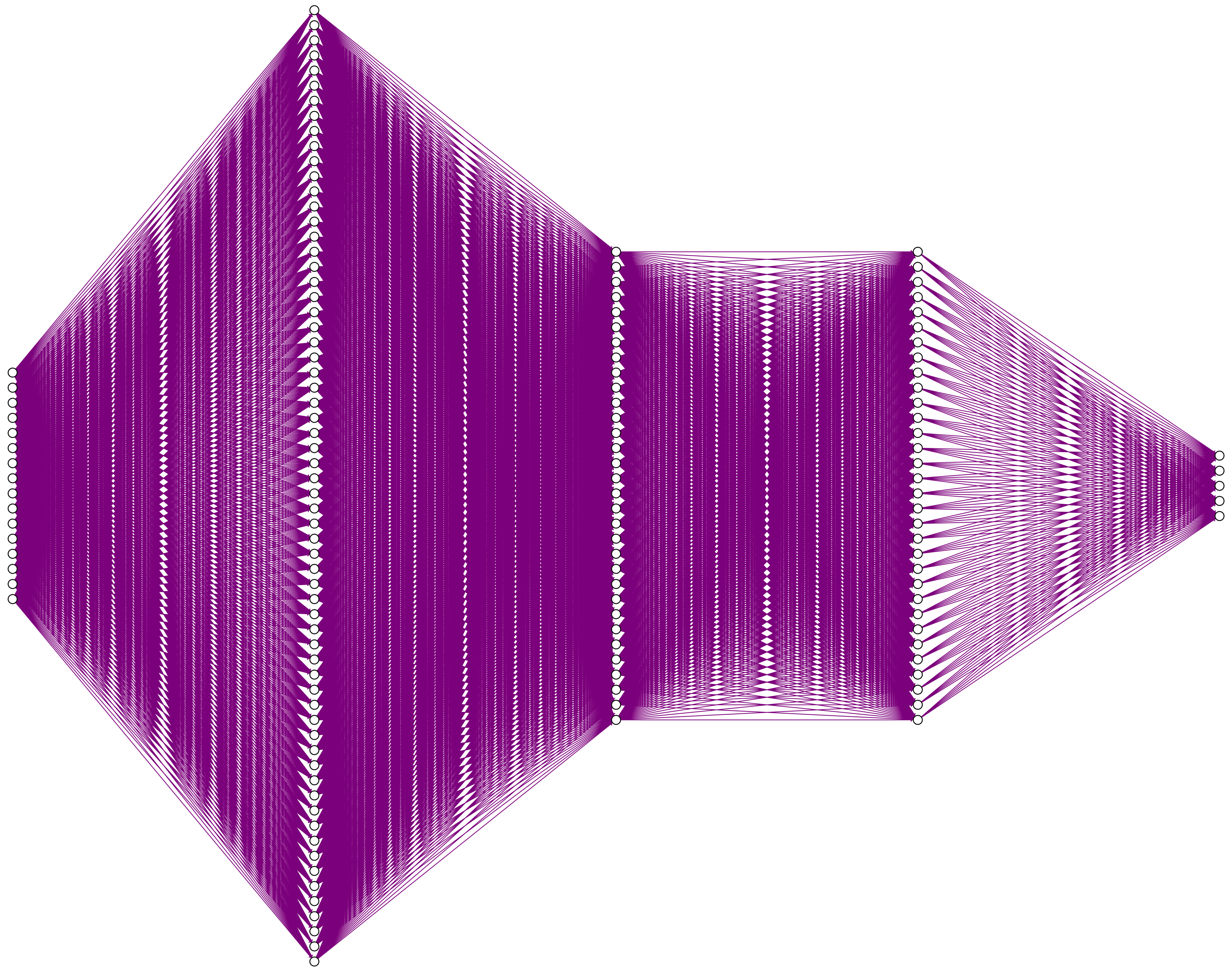
ITERATIVE MAGNITUDE-BASED PRUNING

Used in
CMS-DP-2022-020
for NN vertex finder

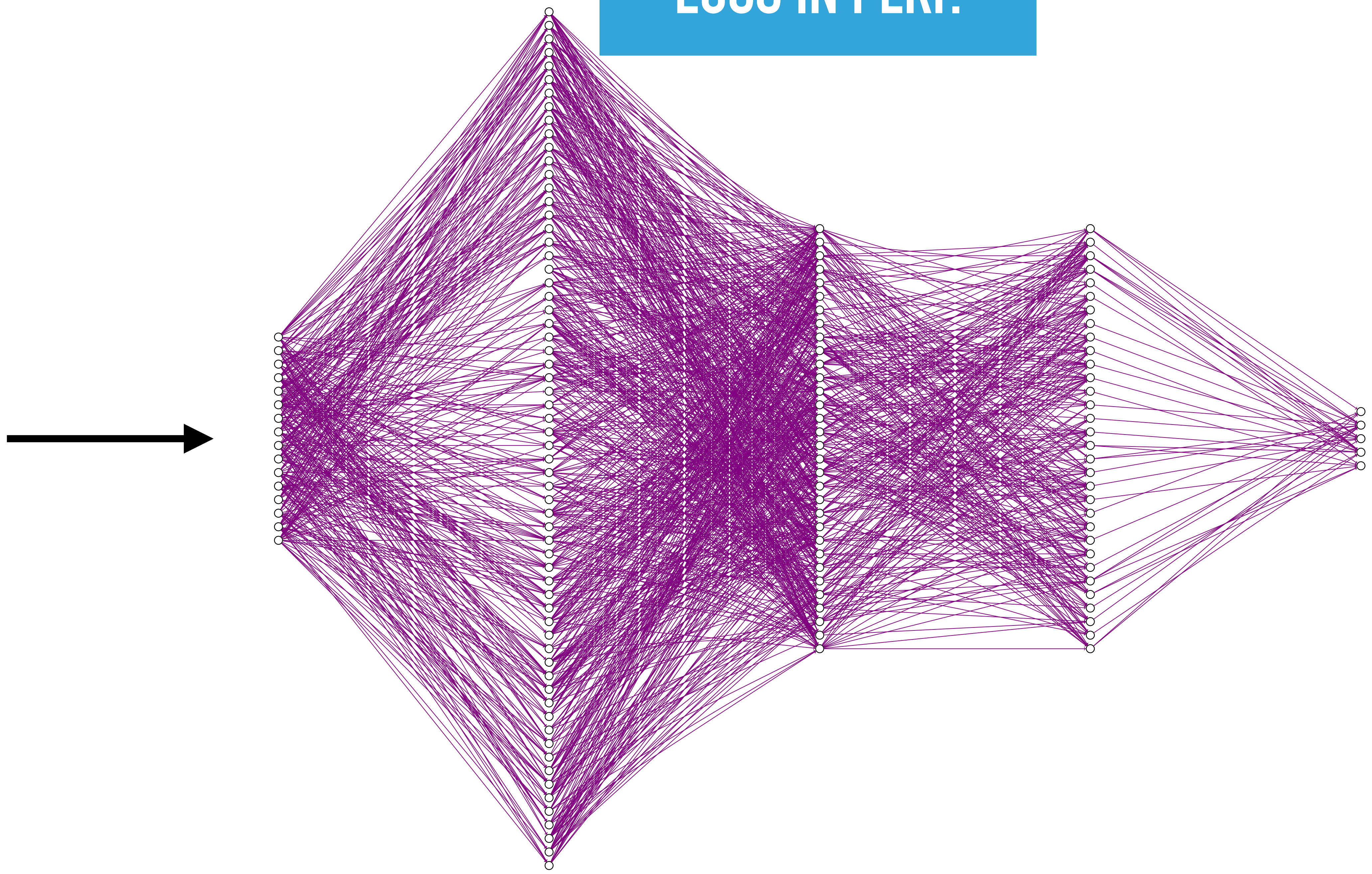
- Train with **L_1 regularization** (down-weights unimportant synapses)

$$L_\lambda(\boldsymbol{w}) = L(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_1 \qquad \|\boldsymbol{w}\|_1 = \sum_i |w_i|$$

- Remove **smallest** weights
- Iterate



70% REDUCTION OF WEIGHTS WITH NO LOSS IN PERF.



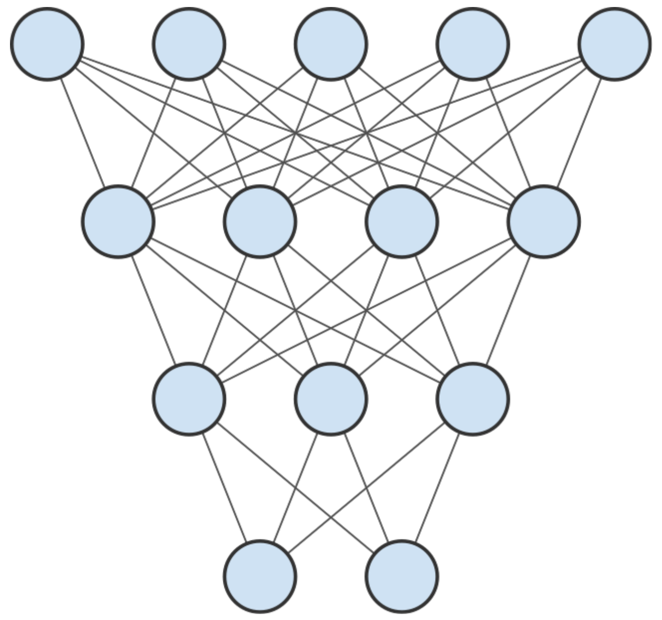
KNOWLEDGE DISTILLATION

KNOWLEDGE DISTILLATION

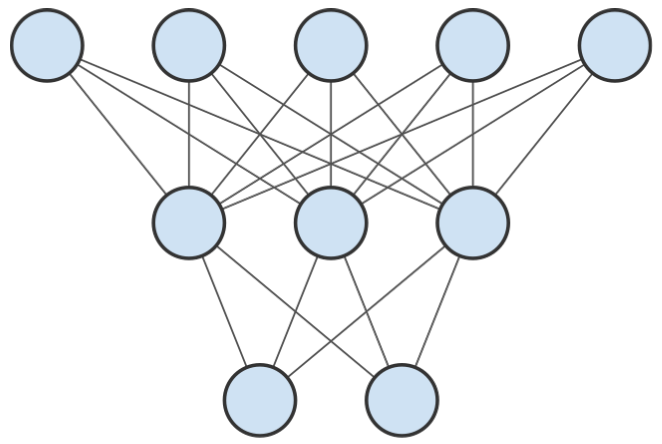
- Can we compress the architecture as well?

KNOWLEDGE DISTILLATION

- Can we compress the architecture as well?
- Knowledge distillation: training a small **student network** to emulate a larger **teacher model** or ensemble of networks



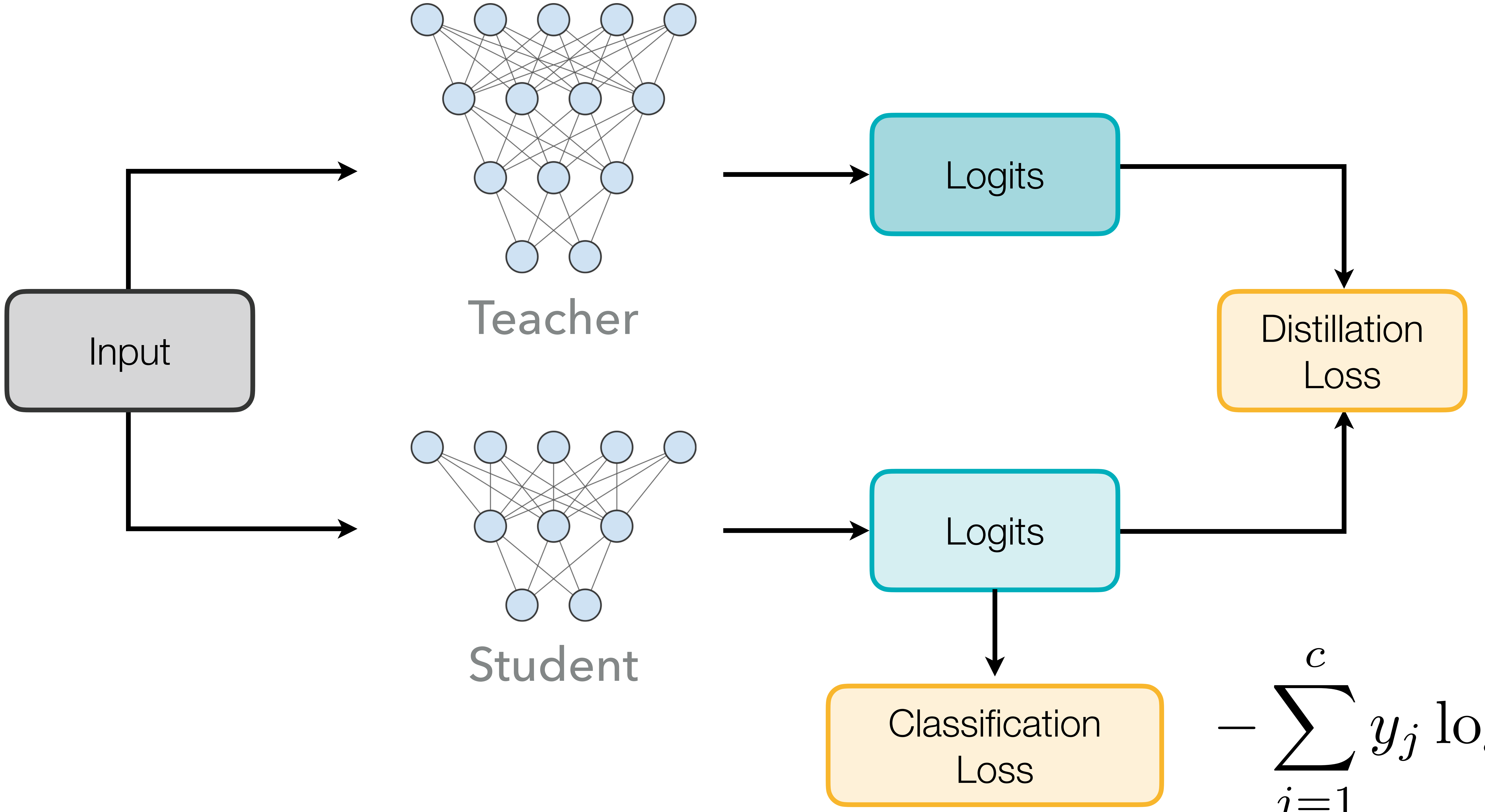
Teacher



Student

KNOWLEDGE DISTILLATION

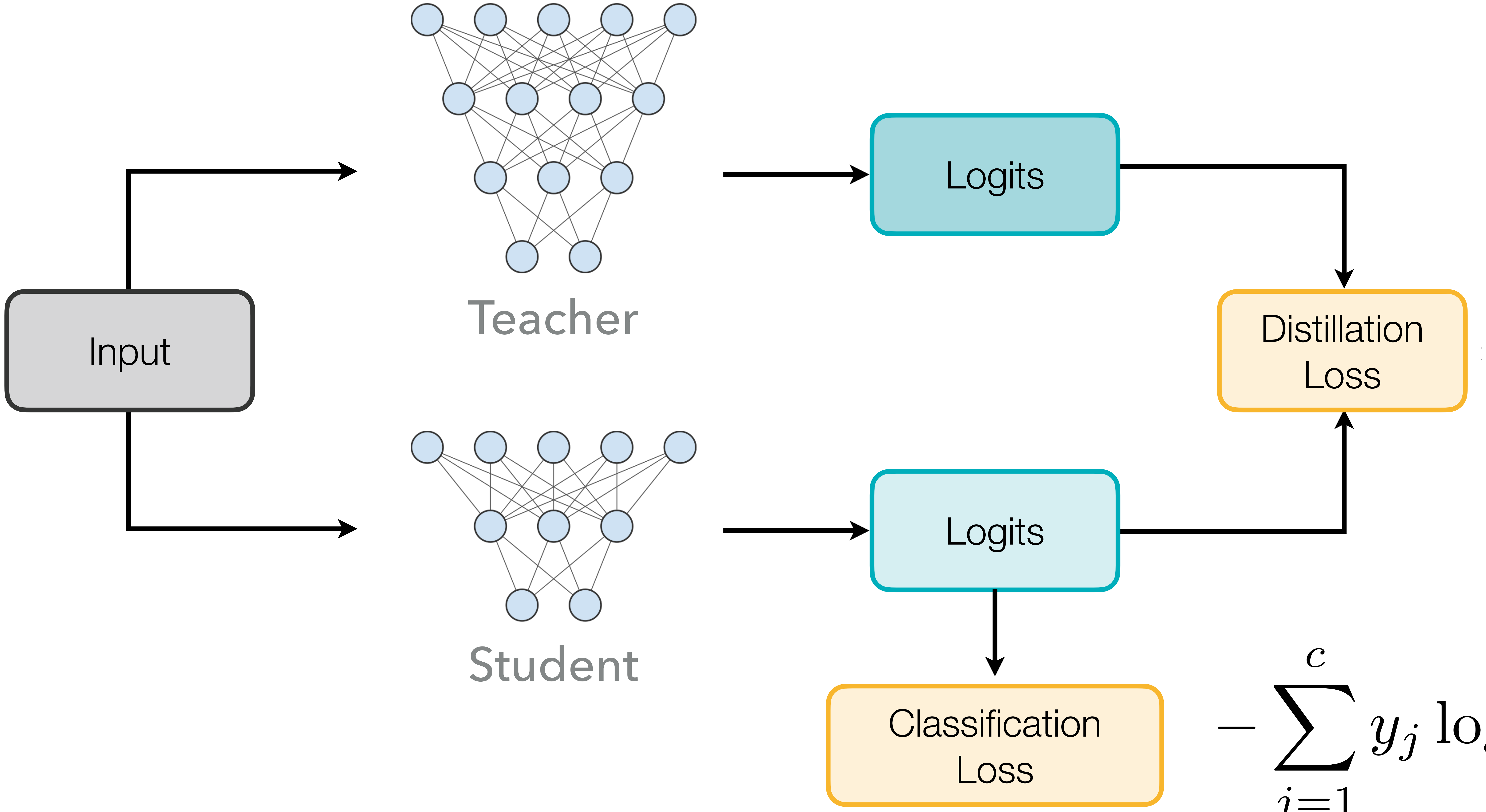
- Can we compress the architecture as well?
- Knowledge distillation: training a small **student network** to emulate a larger **teacher model** or ensemble of networks



$$-\sum_{j=1}^{c} y_j \log \sigma_j(\mathbf{z}_s)$$

KNOWLEDGE DISTILLATION

- Can we compress the architecture as well?
- Knowledge distillation: training a small **student network** to emulate a larger **teacher model** or ensemble of networks



Temperature τ smoothes teacher's distribution (easier to learn from)

$$-\tau^2 \sum_{j=1}^{c} \sigma_j\left(\frac{\mathbf{z}_t}{\tau}\right) \log \sigma_j\left(\frac{\mathbf{z}_s}{\tau}\right)$$

$$-\sum_{j=1}^{c} y_j \log \sigma_j(\mathbf{z}_s)$$

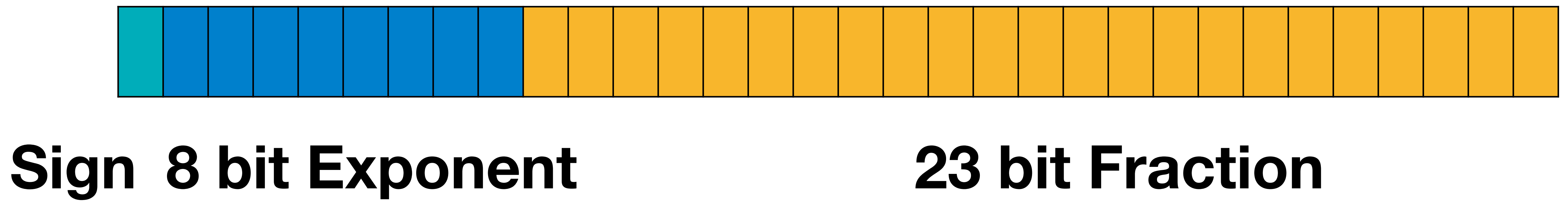
QUANTIZATION TYPES

QUANTIZATION TYPES

- Quantization: using reduced precision for parameters and operations

QUANTIZATION TYPES

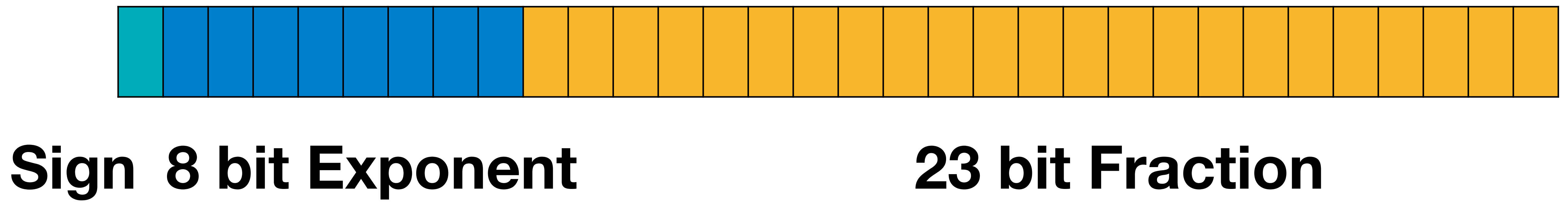
- Quantization: using reduced precision for parameters and operations
  - Baseline: 32-bit floating-point precision



QUANTIZATION TYPES

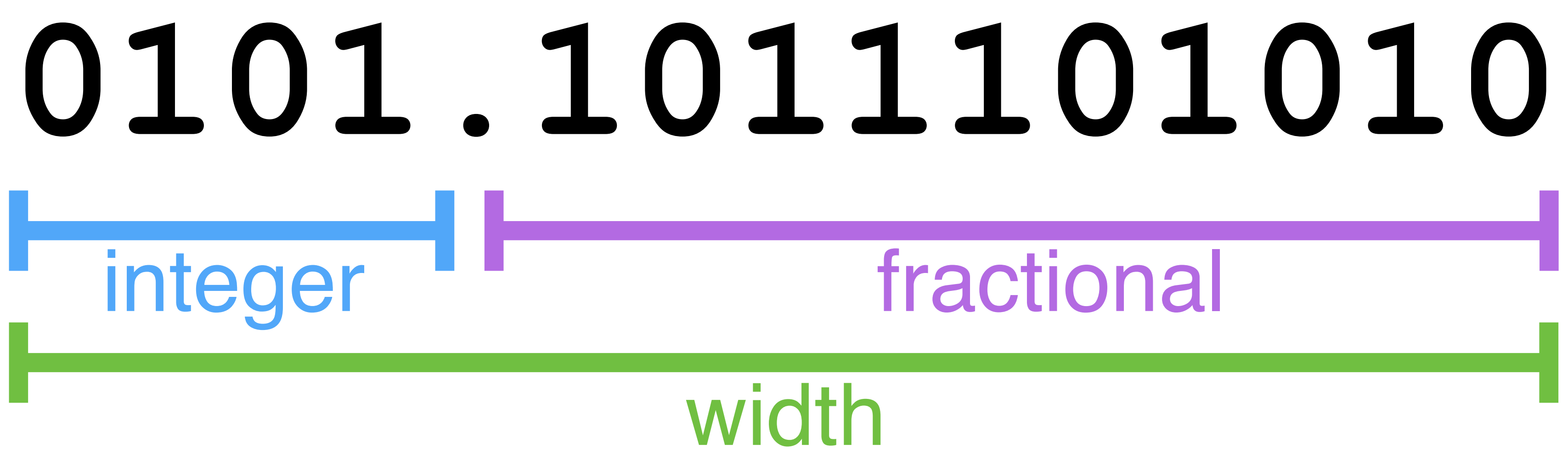
arXiv:2004.09602

- Quantization: using reduced precision for parameters and operations
  - Baseline: 32-bit floating-point precision



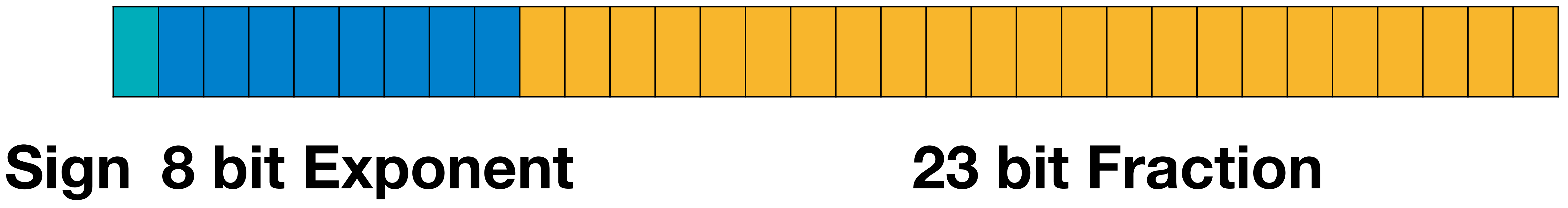
- Fixed-point precision

ap_fixed<width,integer>

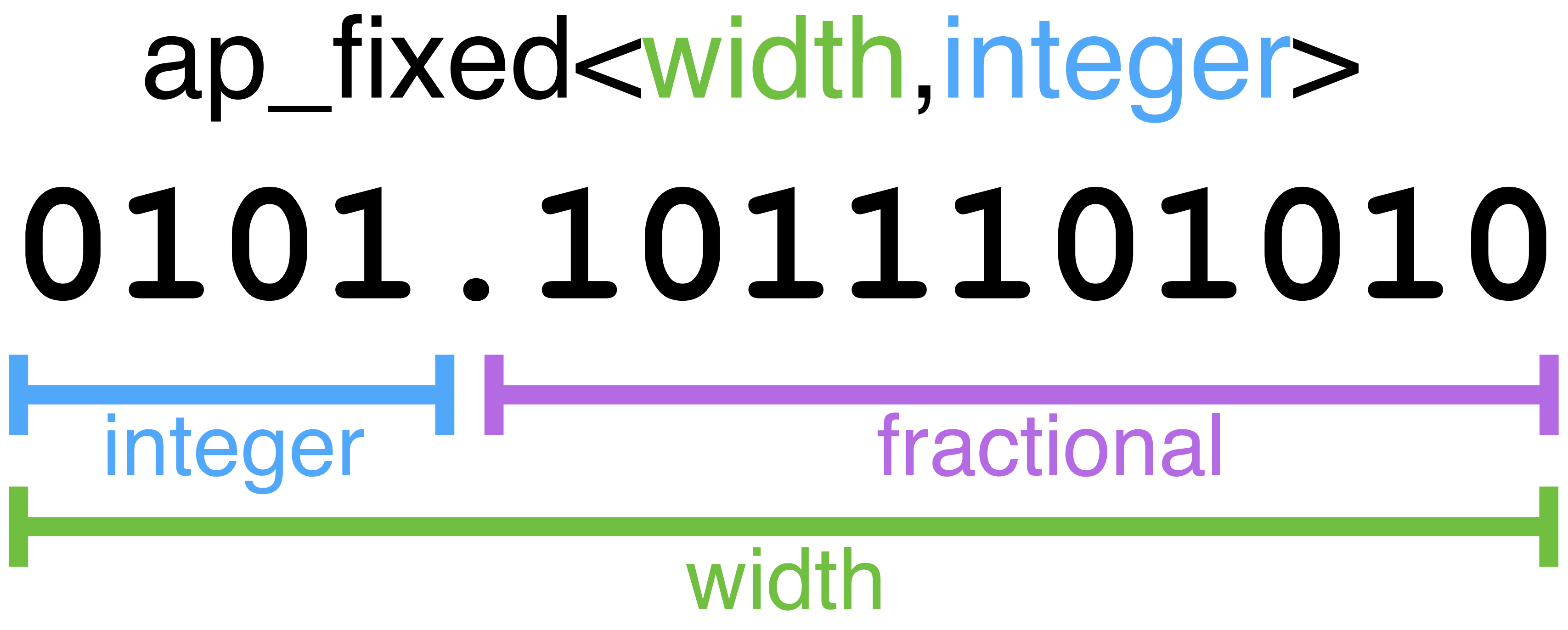


QUANTIZATION TYPES

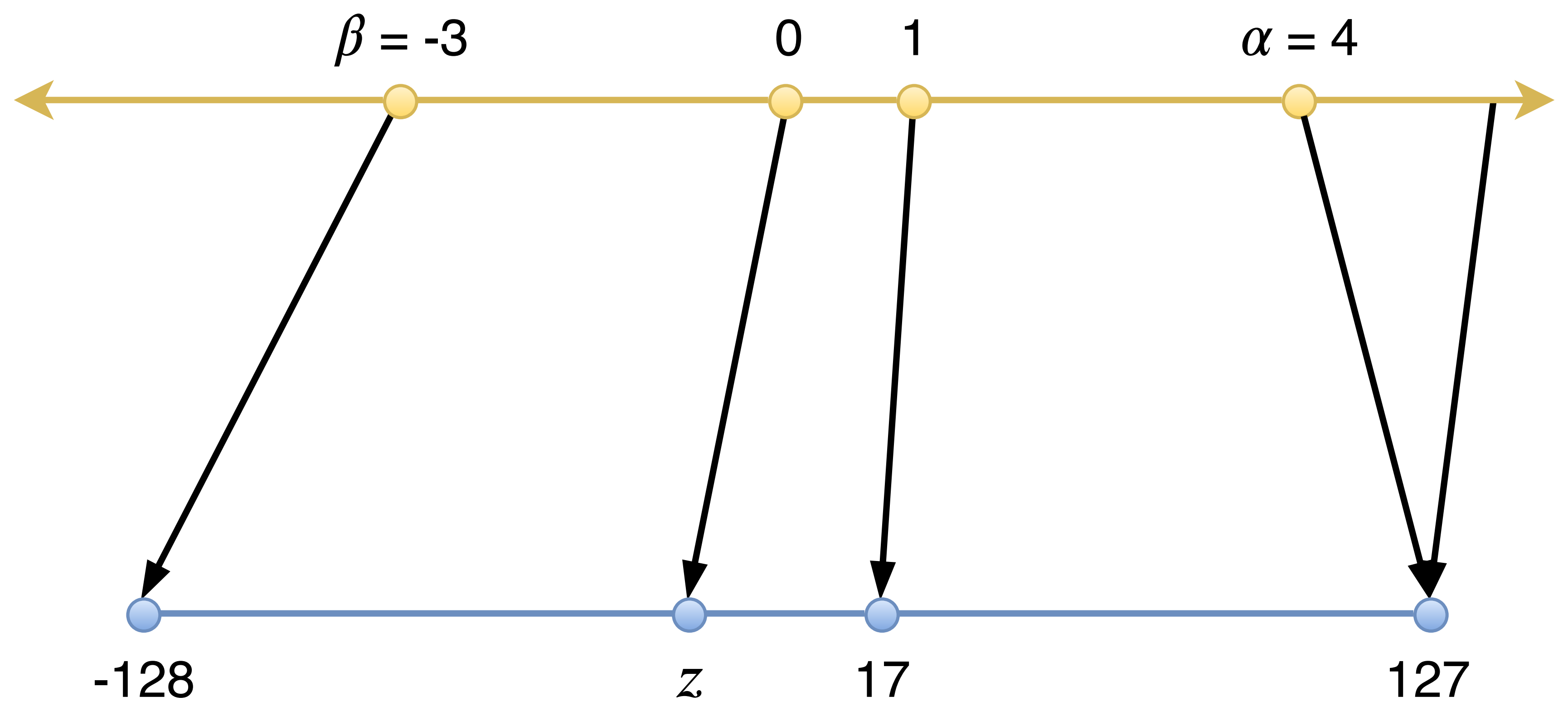
- Quantization: using reduced precision for parameters and operations
  - Baseline: 32-bit floating-point precision



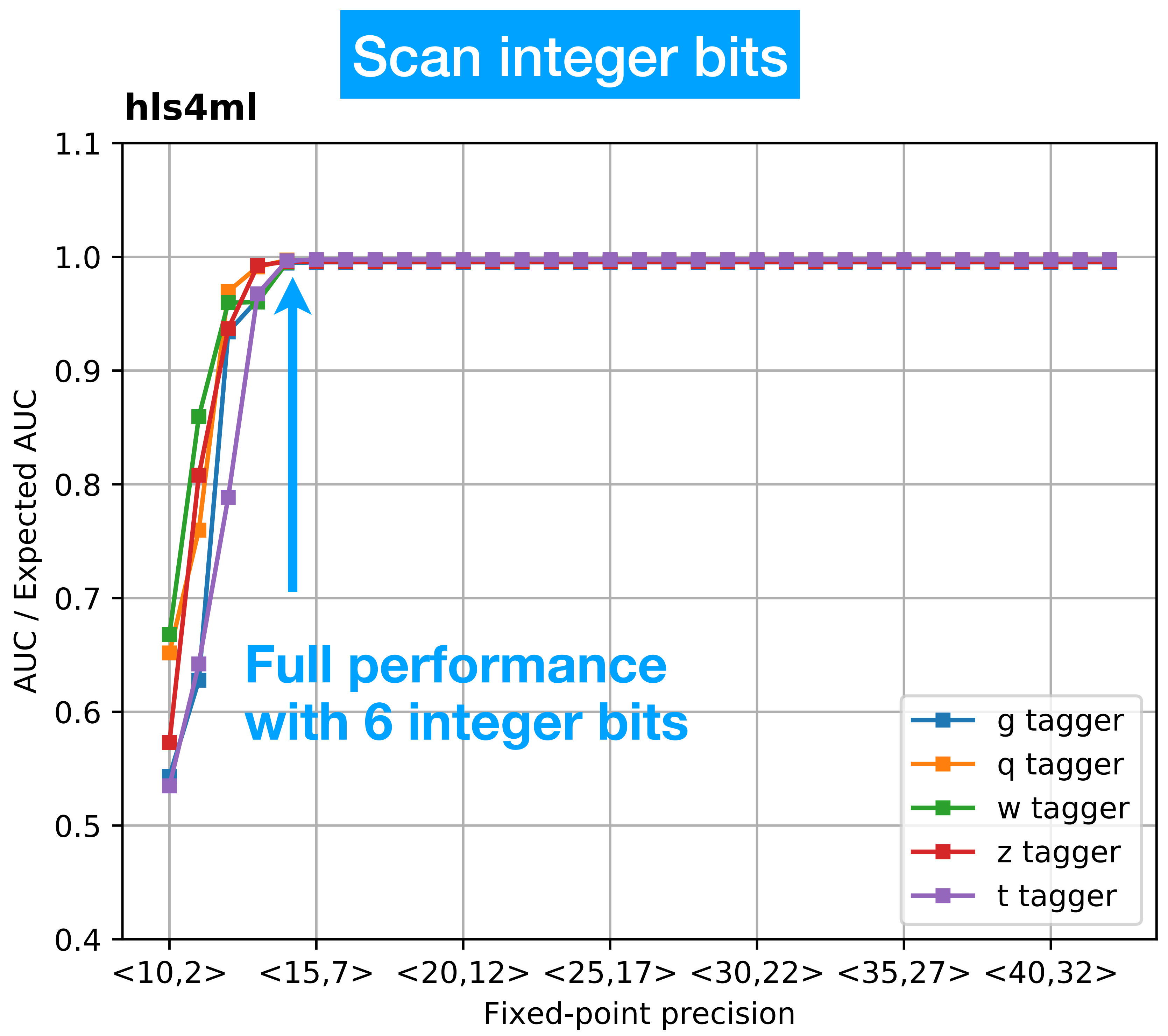
- Fixed-point precision



- Affine integer quantization



POST-TRAINING QUANTIZATION



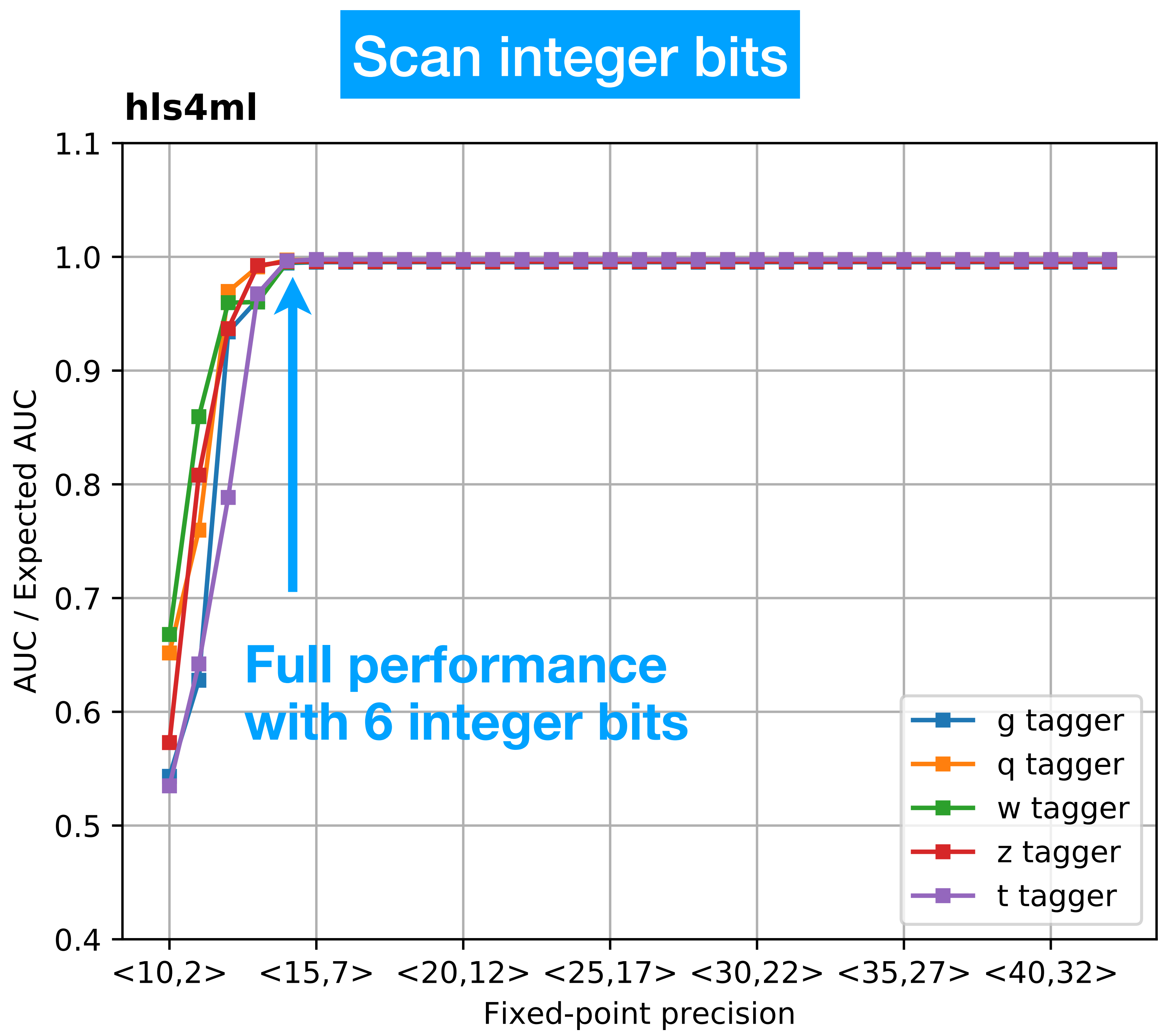
ap_fixed<width,integer>

0101.1011101010

integer fractional

width

POST-TRAINING QUANTIZATION



- General strategy: avoid overflows in integer bit

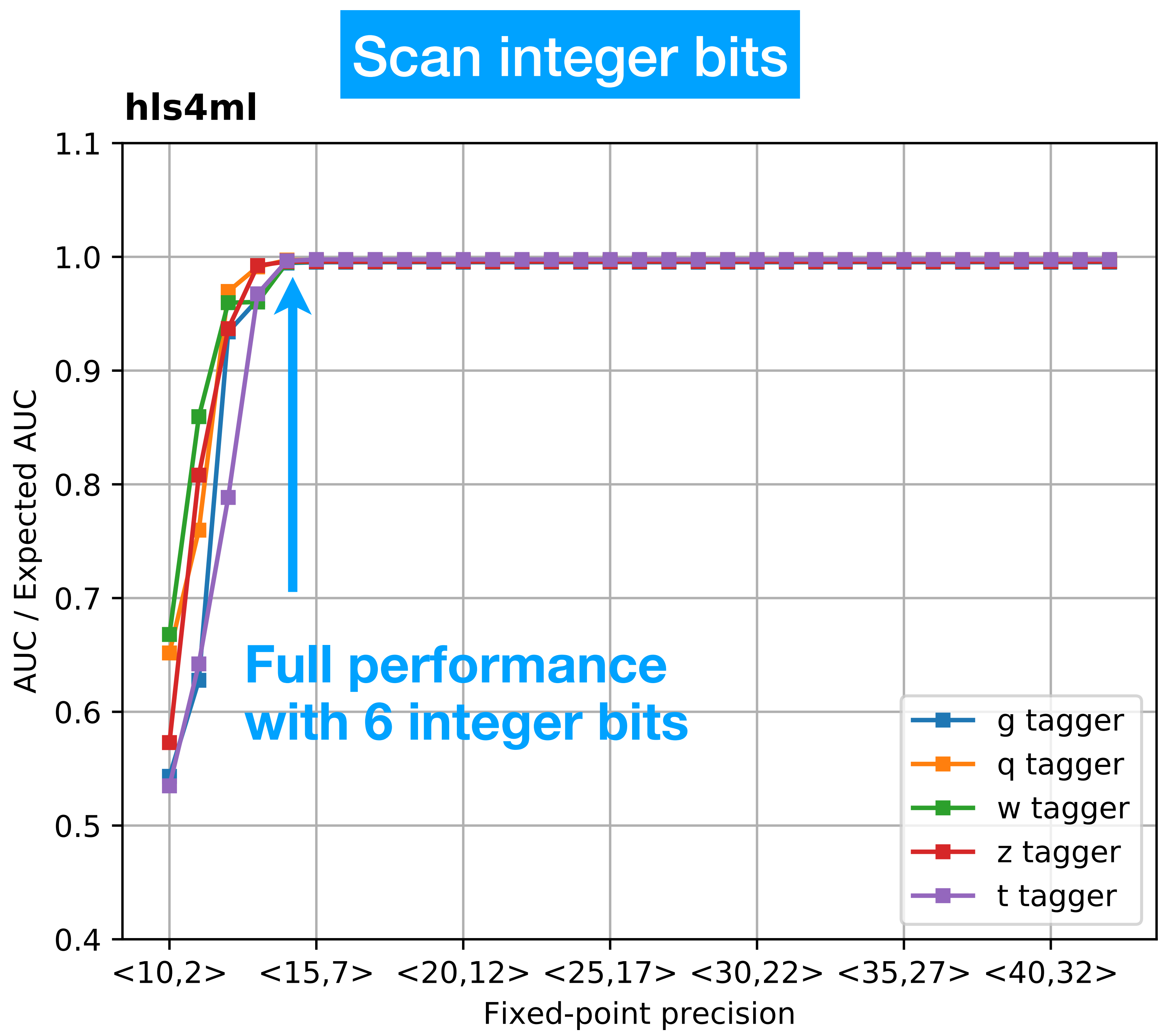
ap_fixed<width,integer>

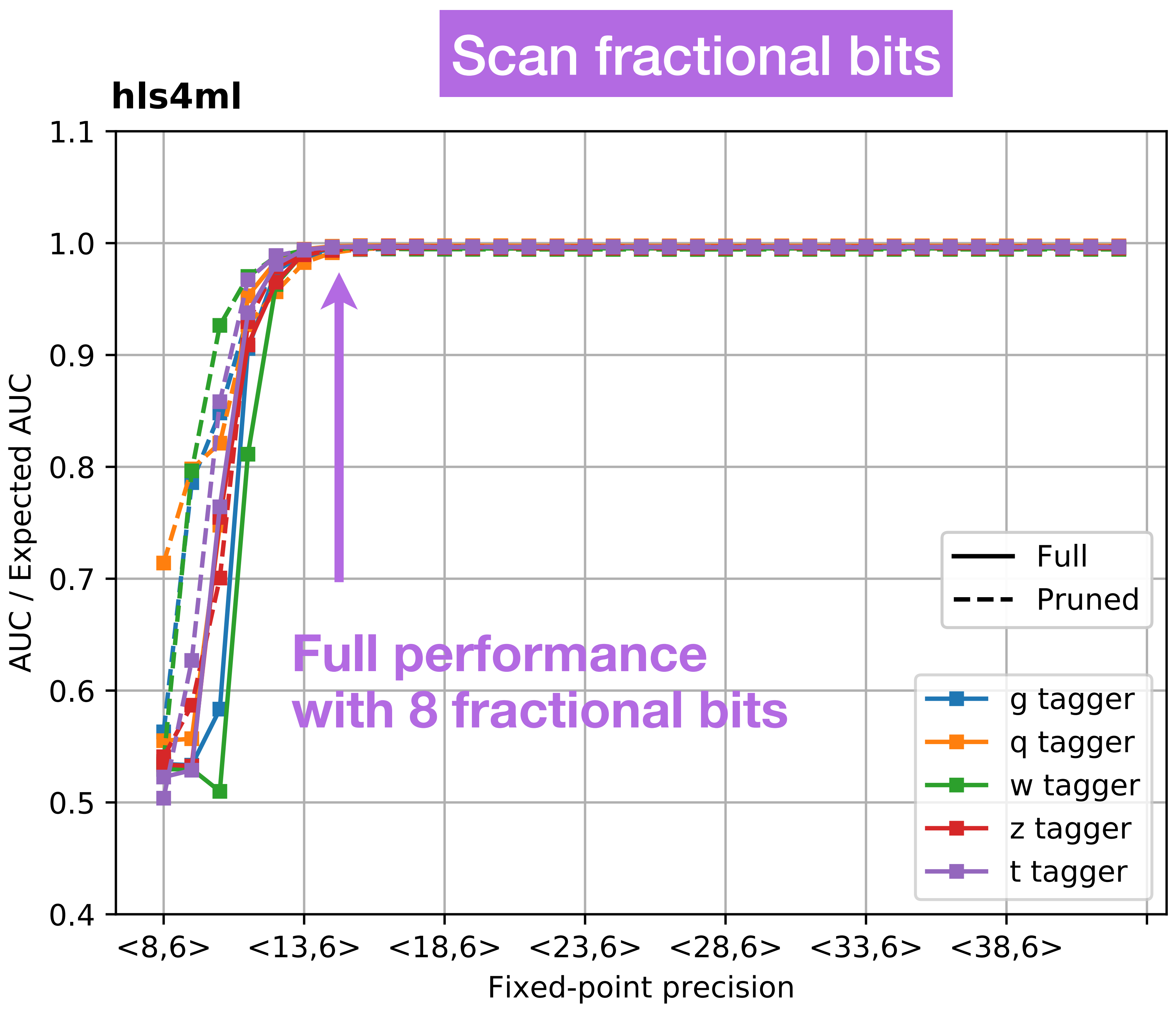
0101.1011101010

integer fractional

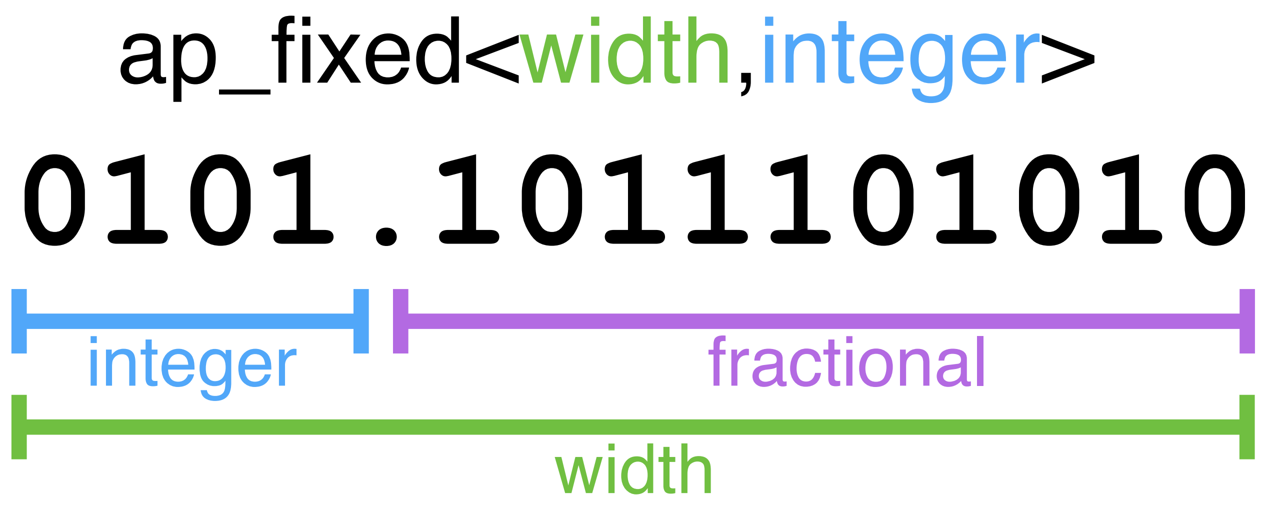
width

POST-TRAINING QUANTIZATION





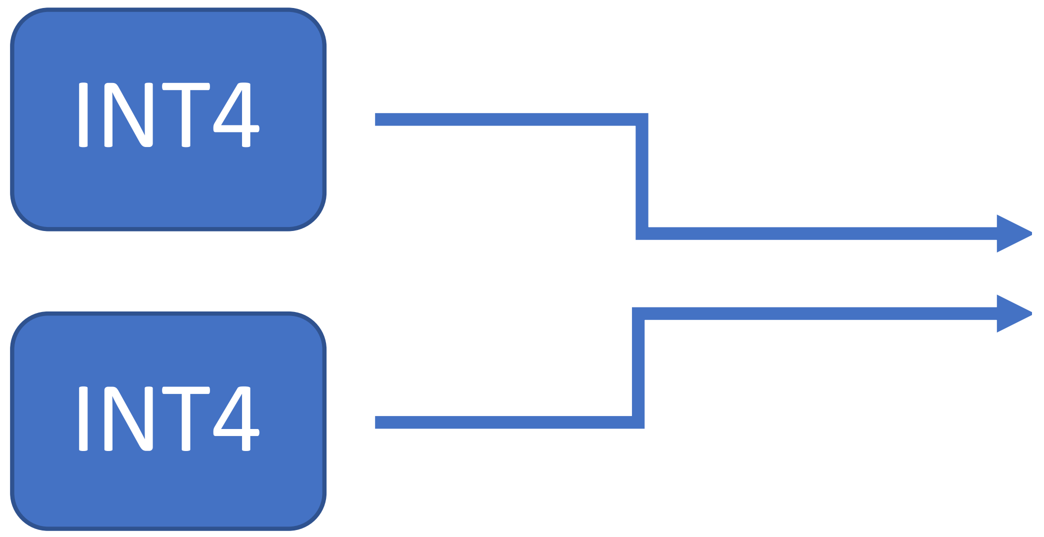
- General strategy: avoid overflows in integer bit
- Then scan the fractional bit width until reaching optimal performance



QUANTIZATION-AWARE TRAINING: HOW DOES IT WORK?

$$W^{\text{fp32}} = (S_w, W^{i4})^{\text{fp32}}$$

Weights



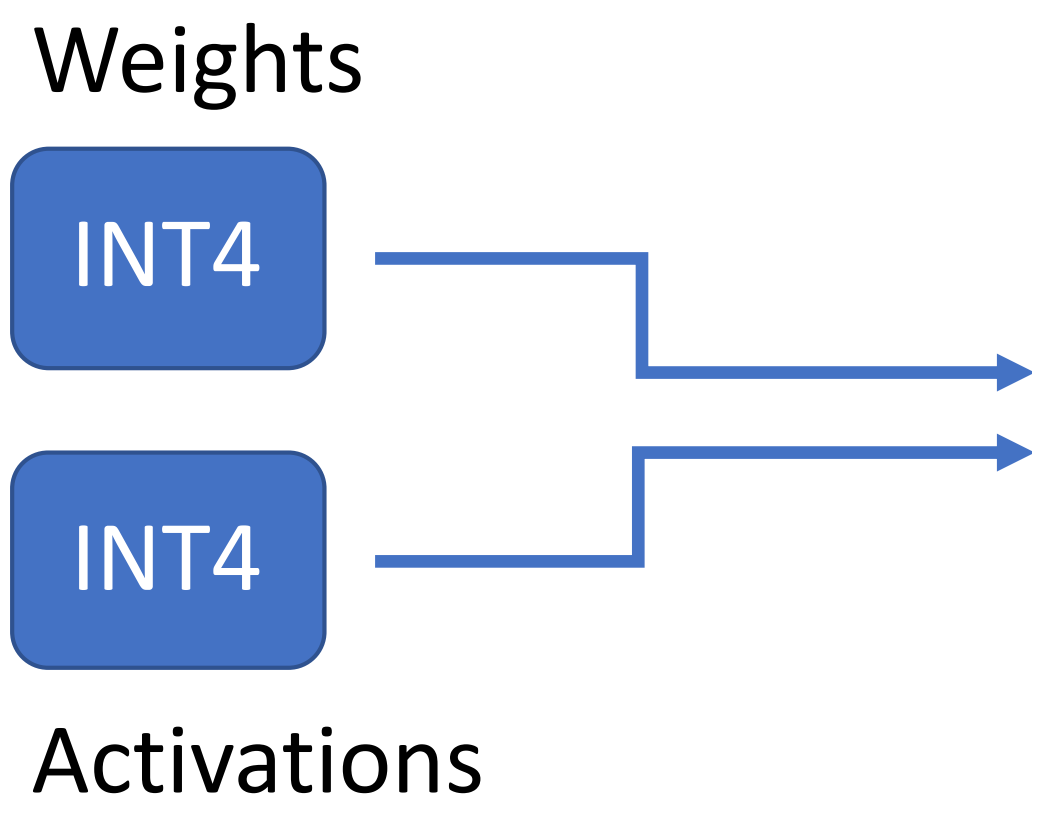
Activations

$$h^{\text{fp32}} = (S_h, h^{i4})^{\text{fp32}}$$

QUANTIZATION-AWARE TRAINING: HOW DOES IT WORK?

- Fake quantization: using 32-bit floating-point under the hood

$$W^{\color{red}{\mathrm{fp32}}} = (S_w, W^{i4})^{\color{red}{\mathrm{fp32}}}$$

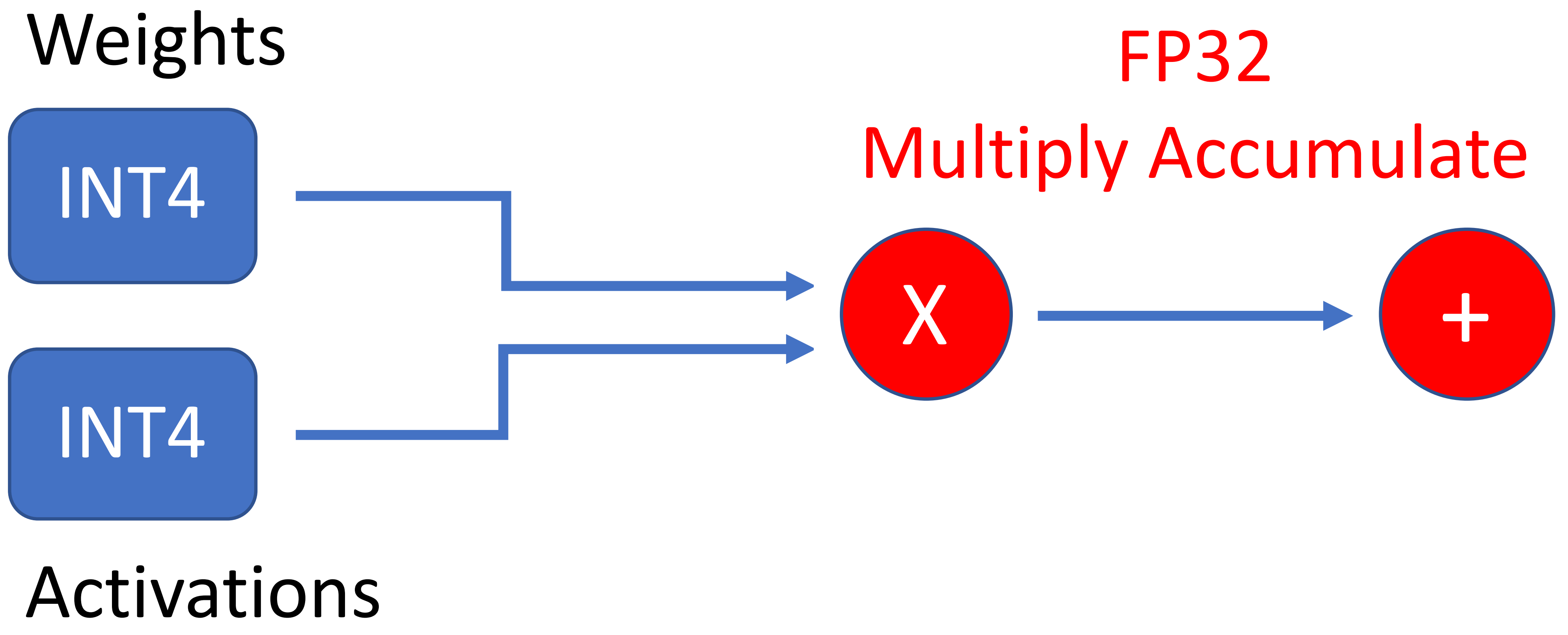


$$h^{\color{red}{\mathrm{fp32}}} = (S_h, h^{i4})^{\color{red}{\mathrm{fp32}}}$$

QUANTIZATION-AWARE TRAINING: HOW DOES IT WORK?

- Fake quantization: using 32-bit floating-point under the hood

$$W^{\text{fp32}} = (S_w, W^{i4})^{\text{fp32}} \qquad a^{\text{fp32}} = W^{\text{fp32}} h^{\text{fp32}}$$



$$h^{\text{fp32}} = (S_h, h^{i4})^{\text{fp32}}$$

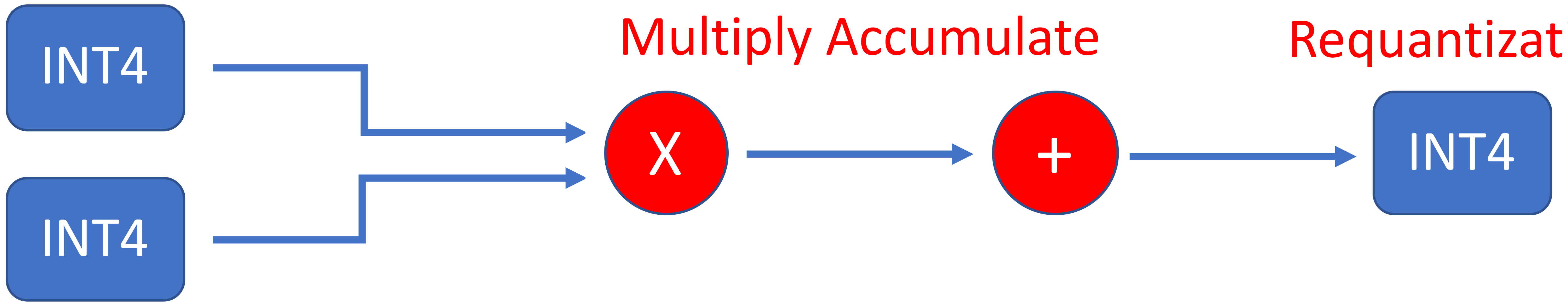
- Fake quantization: using 32-bit floating-point under the hood

$$W^{\text{fp32}} = (S_w, W^{i4})^{\text{fp32}} \qquad a^{\text{fp32}} = W^{\text{fp32}} h^{\text{fp32}} \qquad a^{i4} = Int(\frac{a^{\text{fp32}}}{S_a})$$

Weights

FP32
Multiply Accumulate

FP32 -> INT4
Requantization



Activations

$$h^{\text{fp32}} = (S_h, h^{i4})^{\text{fp32}}$$

QUANTIZATION-AWARE TRAINING: HOW DOES IT WORK?

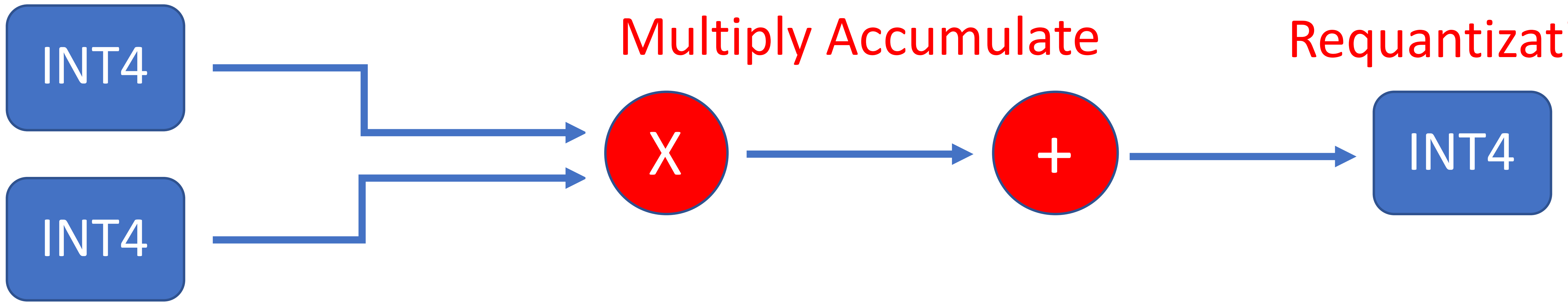
- Fake quantization: using 32-bit floating-point under the hood
- Straight-through estimator: during backpropagation, ignore quantization operation (treat as identity)

$$W^{\text{fp32}} = (S_w, W^{i4})^{\text{fp32}} \qquad a^{\text{fp32}} = W^{\text{fp32}} h^{\text{fp32}} \qquad a^{i4} = Int\left(\frac{a^{\text{fp32}}}{S_a}\right)$$

Weights

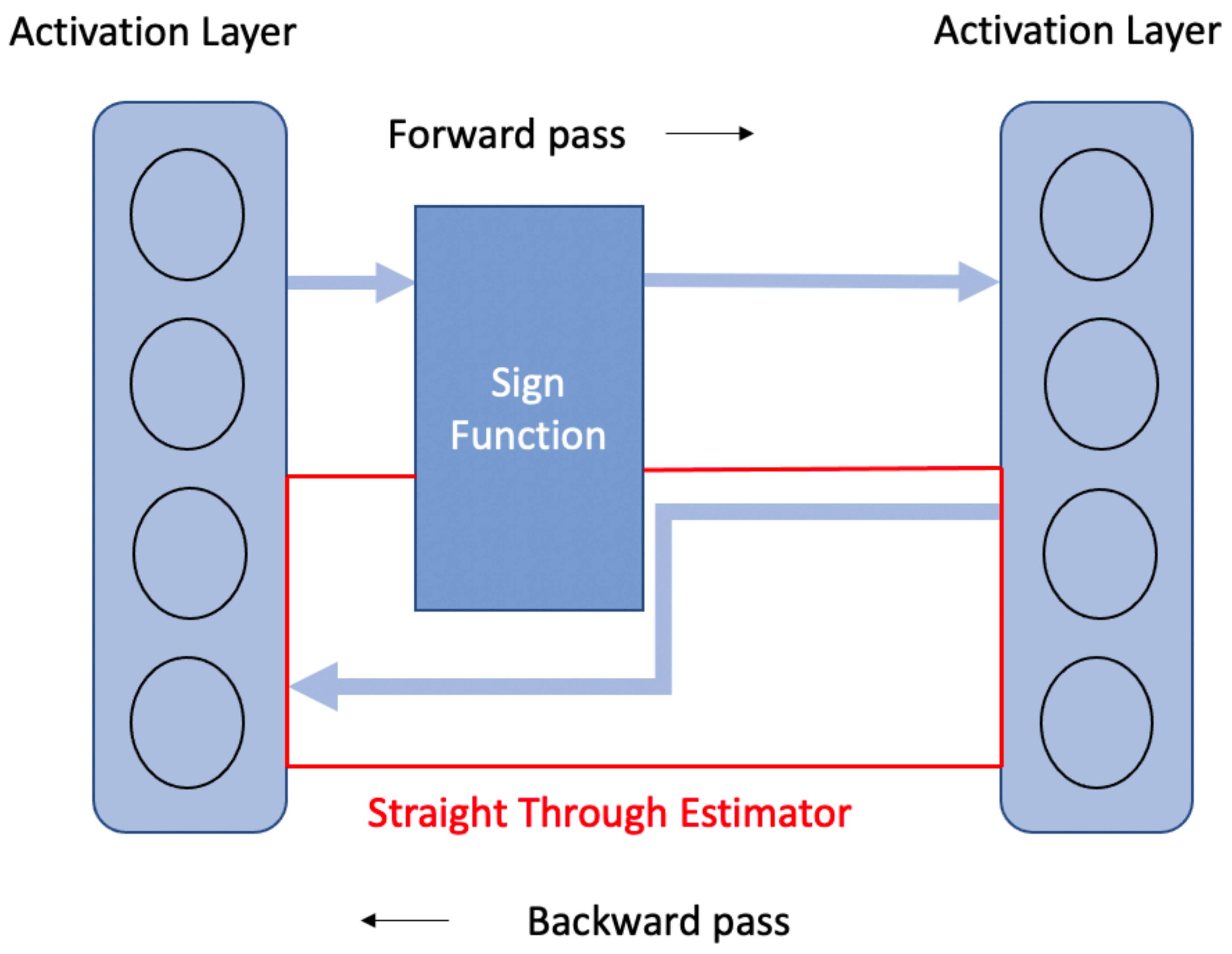
FP32
Multiply Accumulate

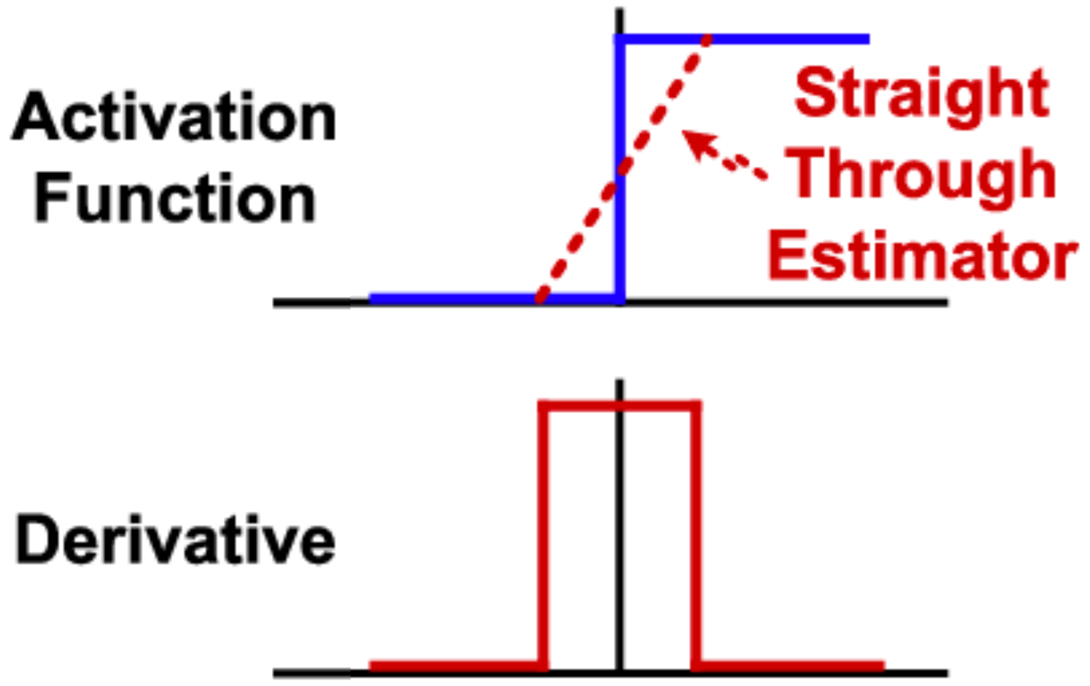
FP32 -> INT4
Requantization



Activations

$$h^{\text{fp32}} = (S_h, h^{i4})^{\text{fp32}}$$

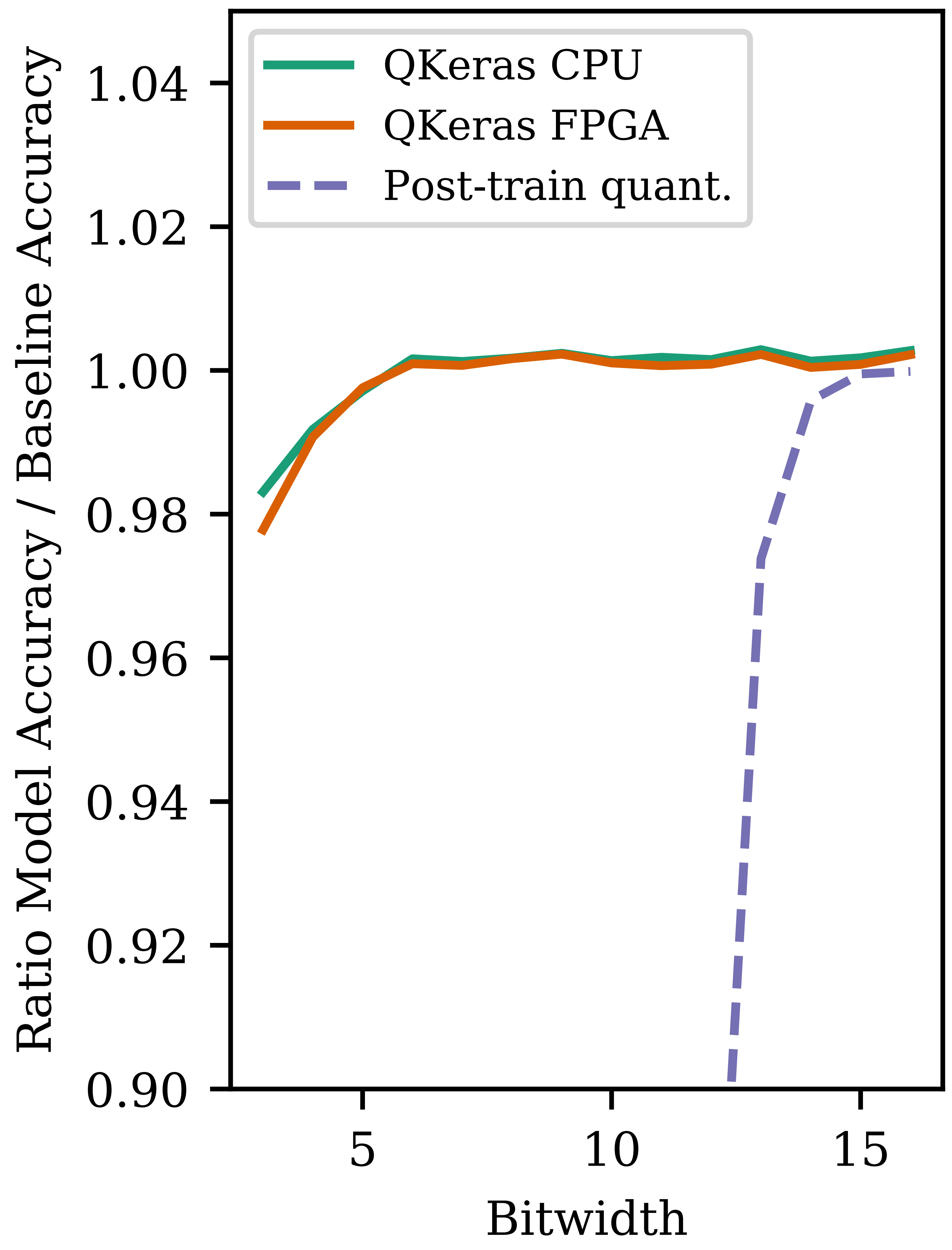




QUANTIZATION-AWARE TRAINING: RESULTS

arXiv:2006.10159

Xilinx VU9P

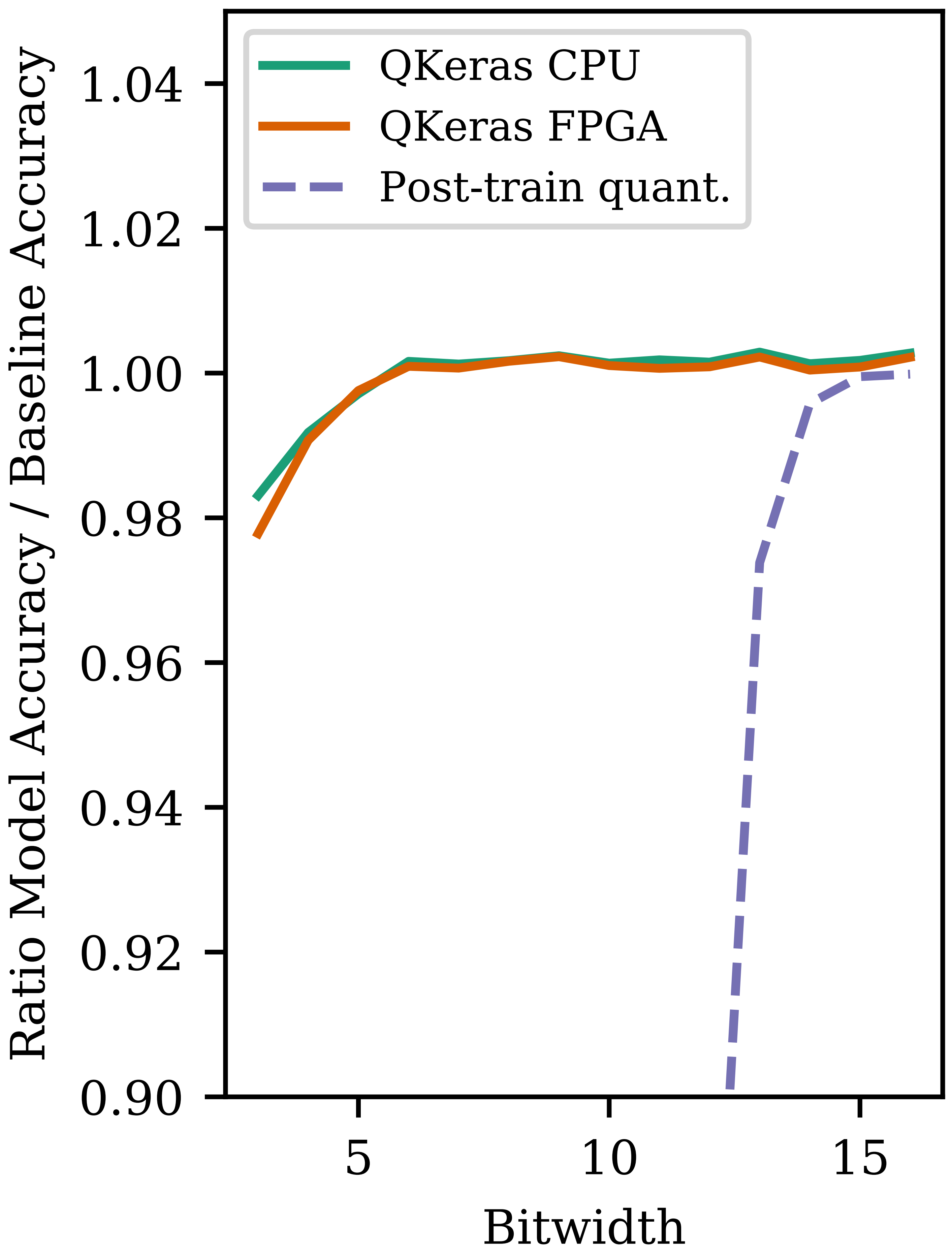


QUANTIZATION-AWARE TRAINING: RESULTS

arXiv:2006.10159

Xilinx VU9P

- Full performance with 6 bits instead of 14 bits

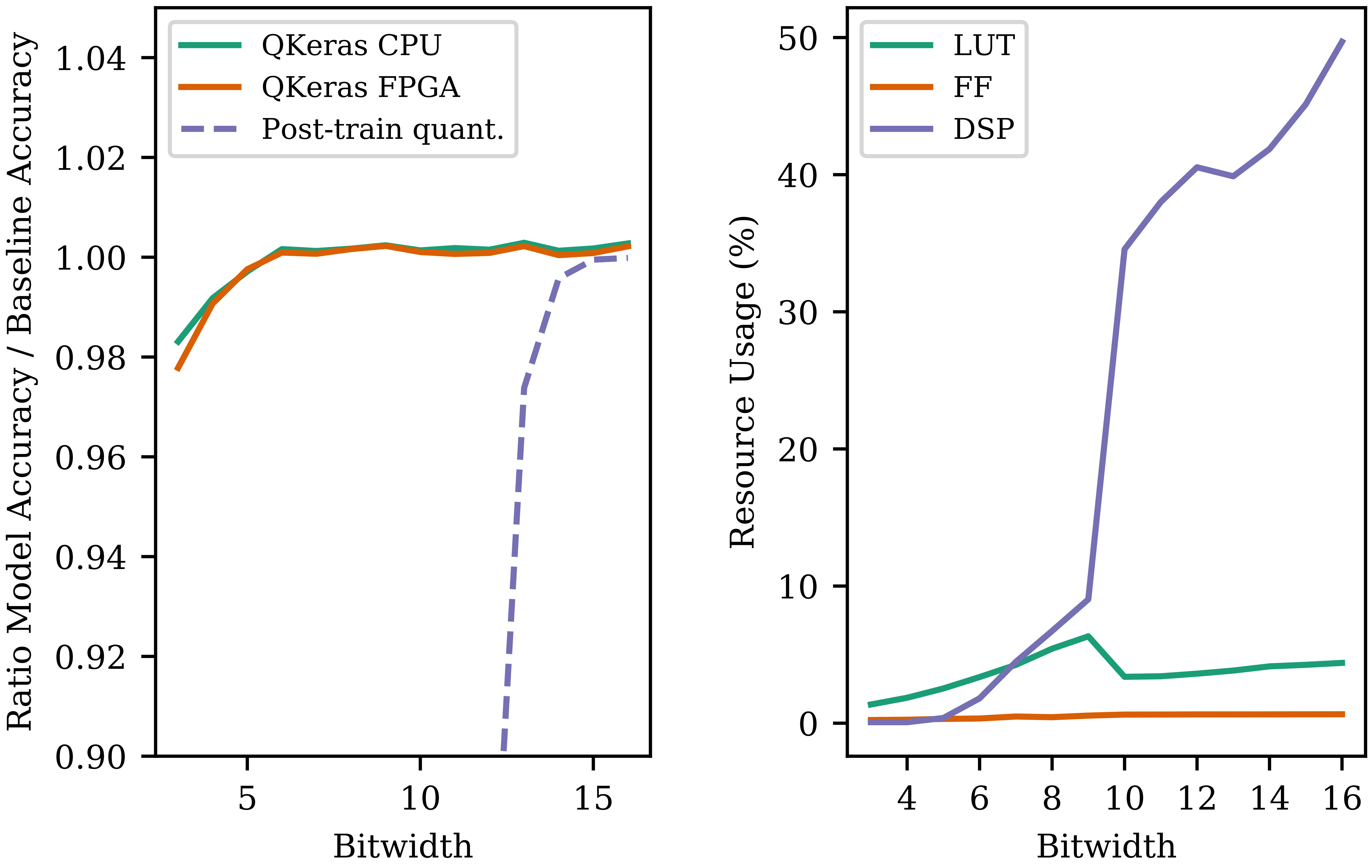


QUANTIZATION-AWARE TRAINING: RESULTS

arXiv:2006.10159

- Full performance with 6 bits instead of 14 bits
- Much smaller fraction of resources

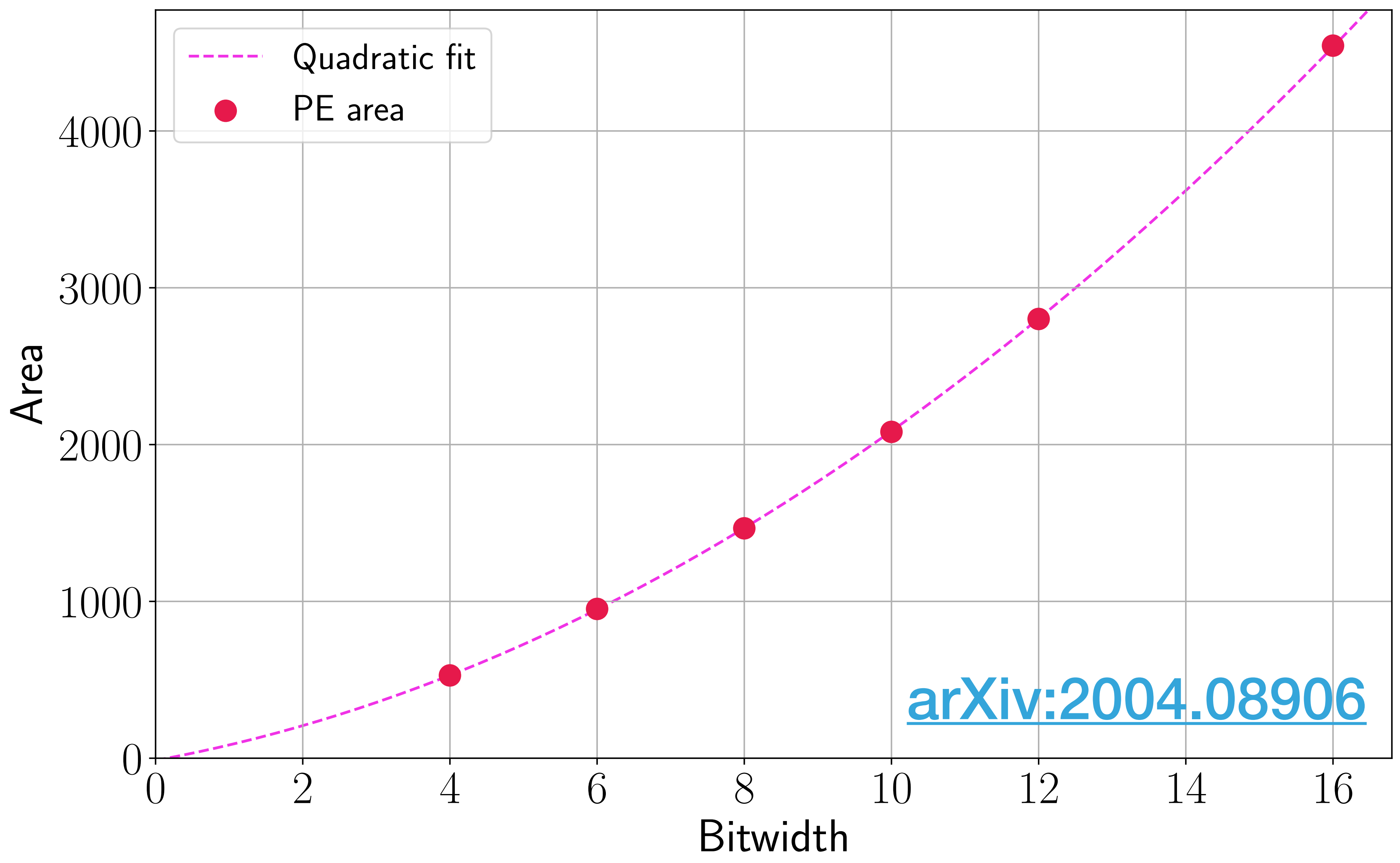
Xilinx VU9P

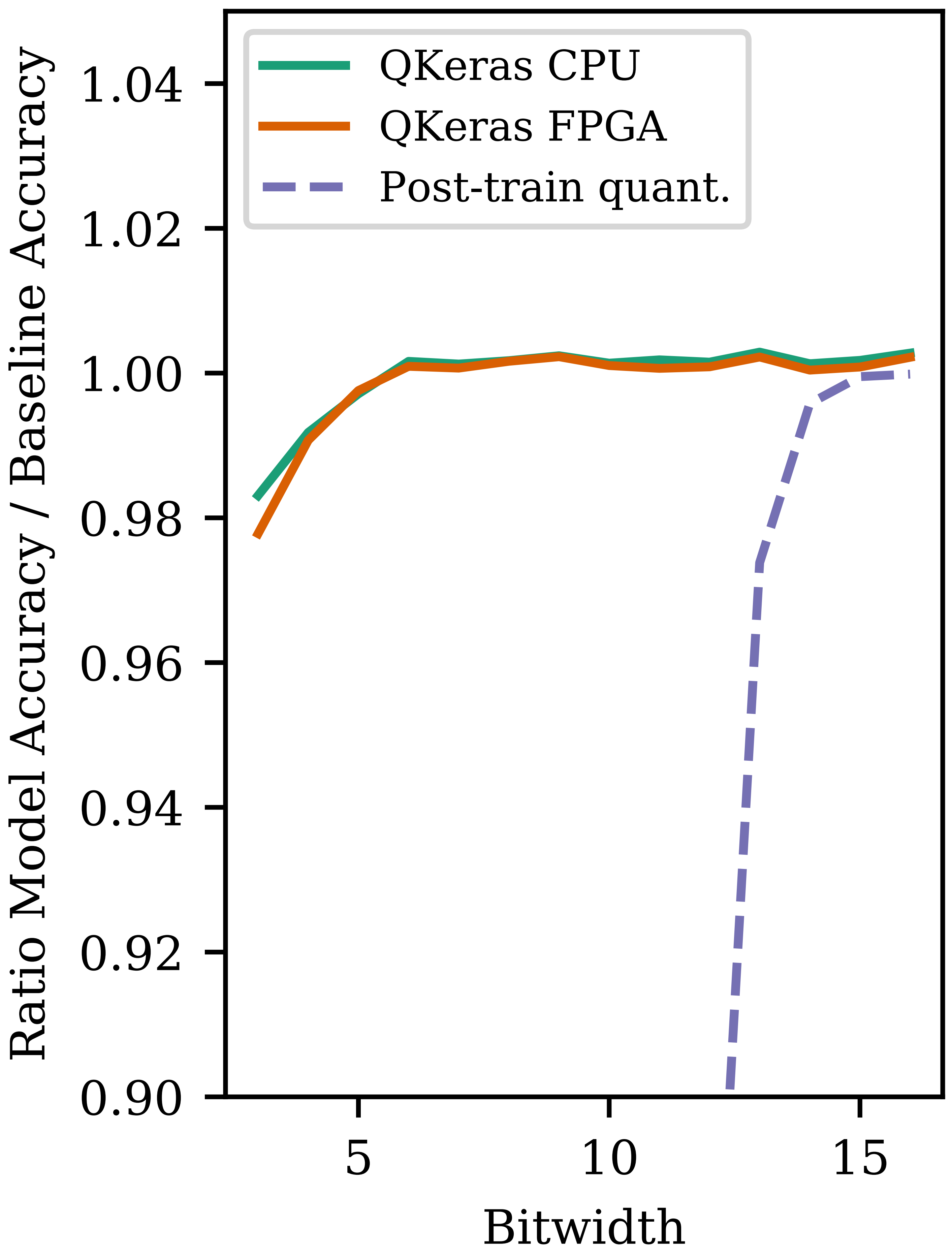


QUANTIZATION-AWARE TRAINING: RESULTS

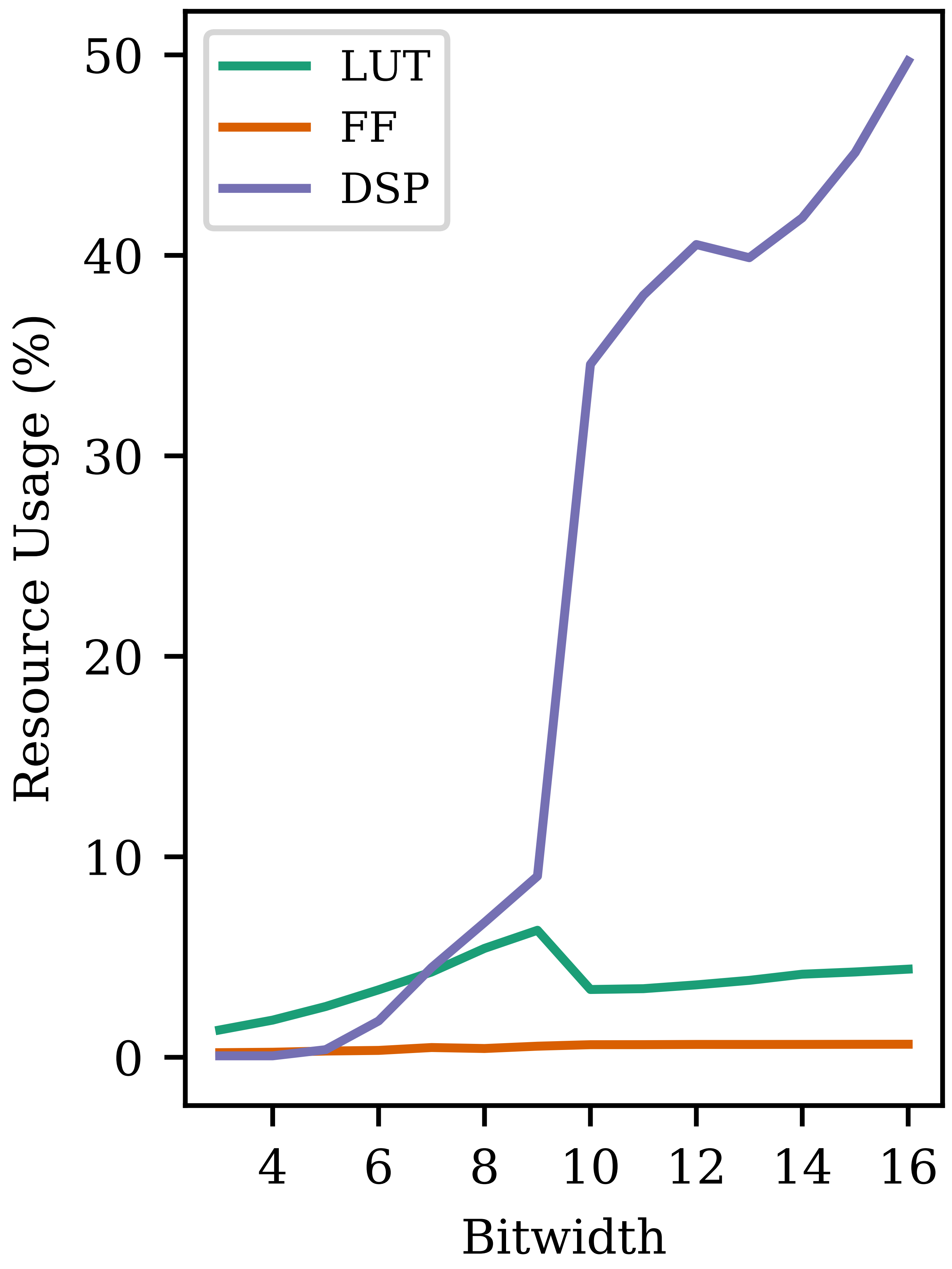
[arXiv:2006.10159](https://arxiv.org/abs/2006.10159)

- Full performance with 6 bits instead of 14 bits
- Much smaller fraction of resources
- Area & power scale quadratically with bit width

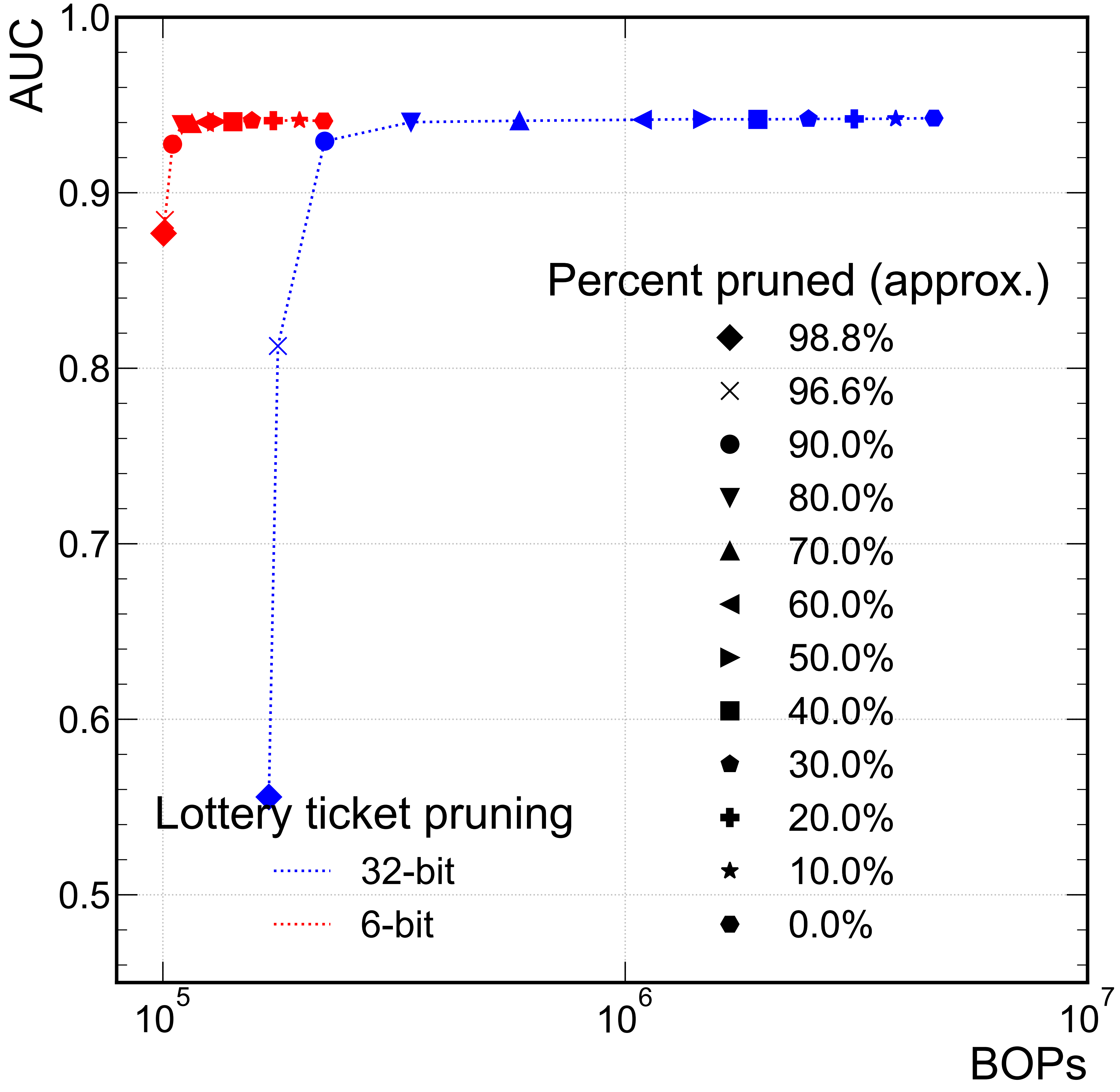




Xilinx VU9P



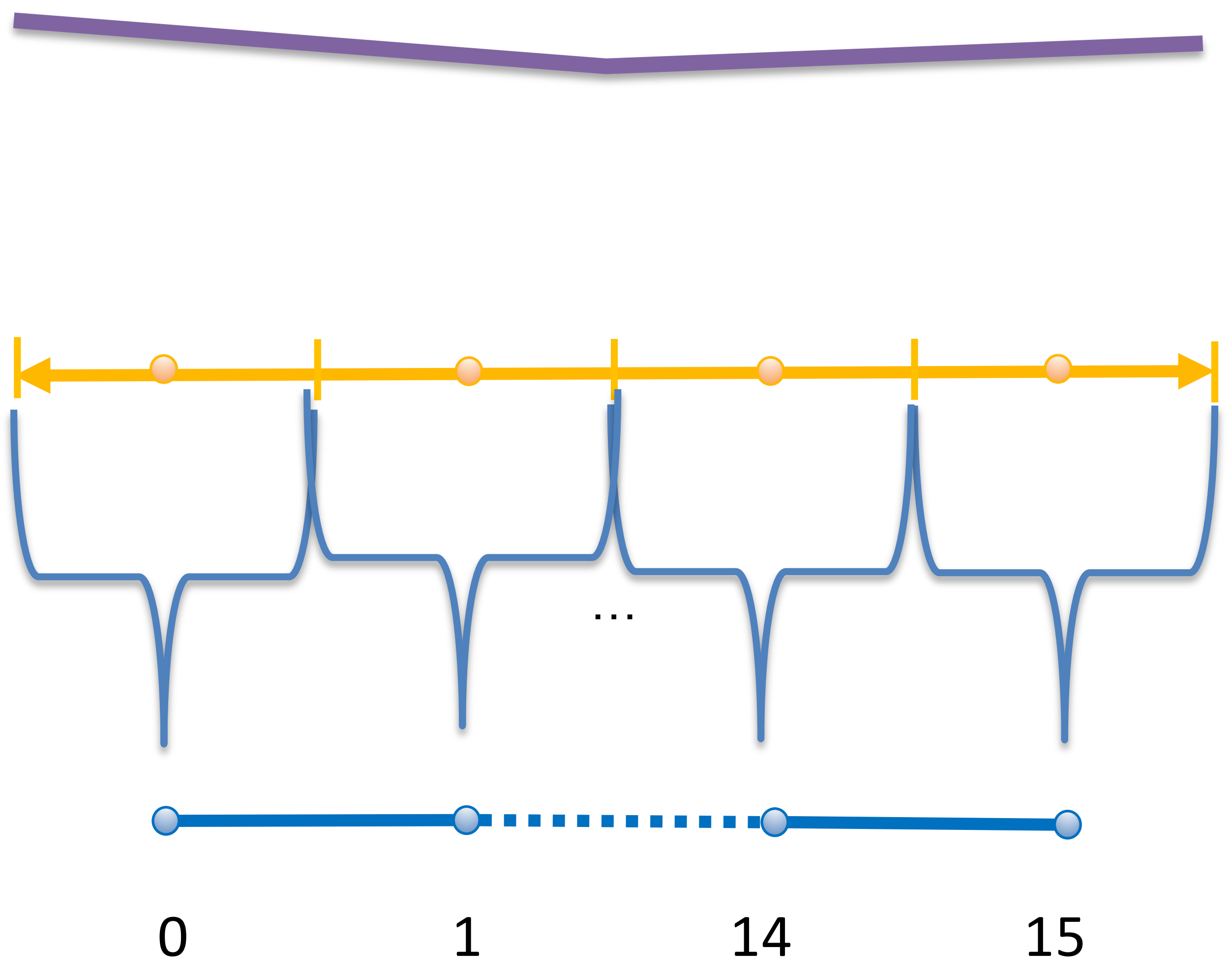
- Quantization-aware pruning (QAP): iterative pruning can further reduce the hardware computational complexity of a quantized model
- After QAP, the 6-bit, 80% pruned model achieves a factor of 50 reduction in BOPs compared to the 32-bit, unpruned model
  - Study using Brevitas



Bit operations (BOPs) definition: arXiv:1804.10969

HESSIAN-AWARE QUANTIZATION (HAWQ)

arXiv:2011.10680



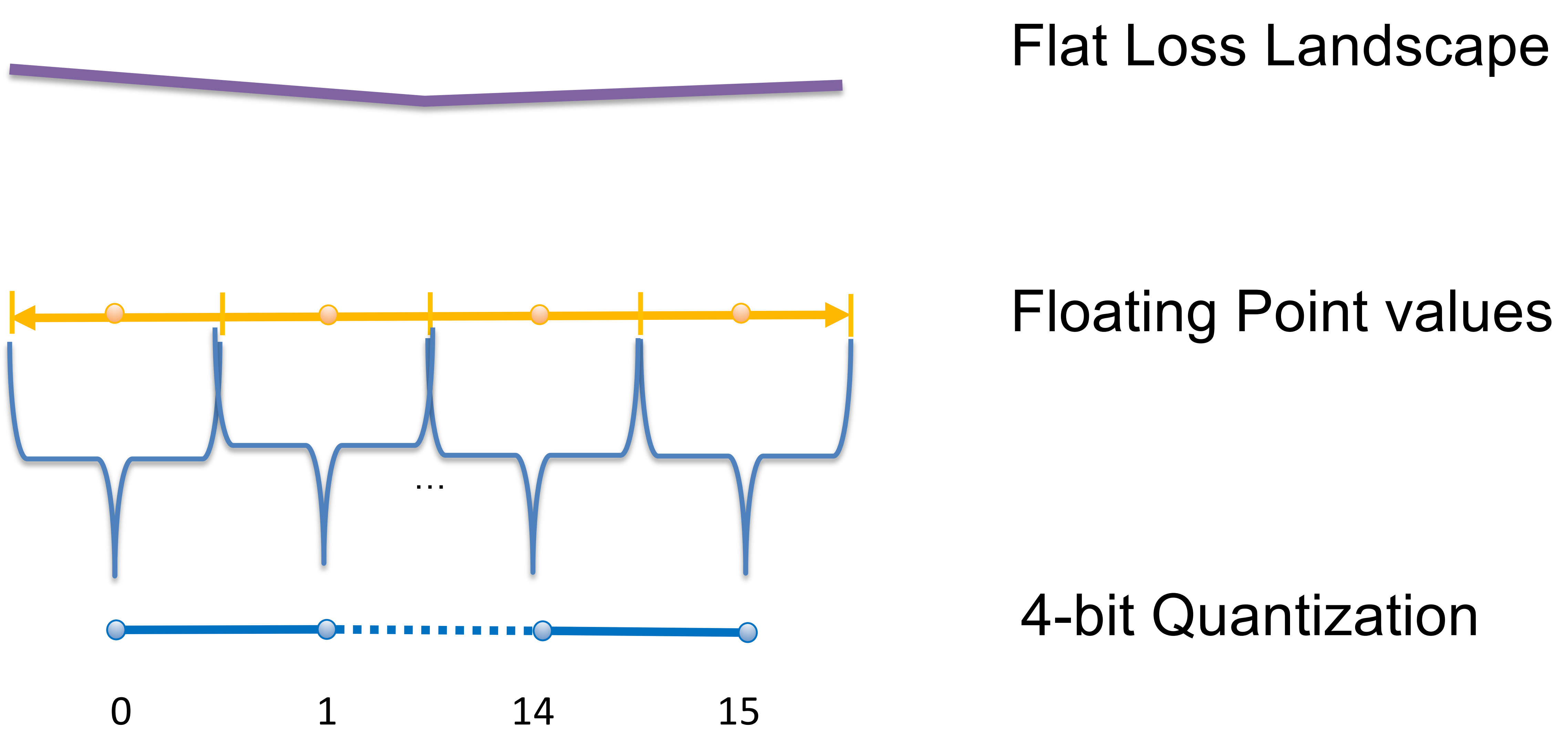
Flat Loss Landscape

Floating Point values

4-bit Quantization

HESSIAN-AWARE QUANTIZATION (HAWQ)

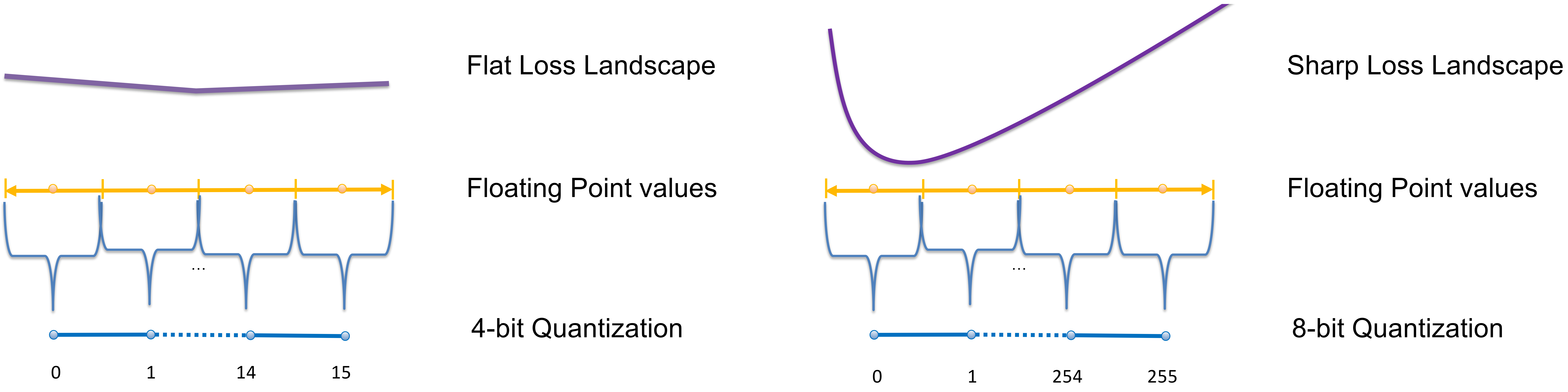
- Hessian of loss can provide additional guidance about quantization!
- Flat loss landscape: Lower bit width



HESSIAN-AWARE QUANTIZATION (HAWQ)

arXiv:2011.10680

- Hessian of loss can provide additional guidance about quantization!
- Flat loss landscape: Lower bit width
- Sharp loss landscape: Higher bit width



I. INTRO & MOTIVATION

II. COMPRESSION

III. HARDWARE

IV. APPLICATIONS

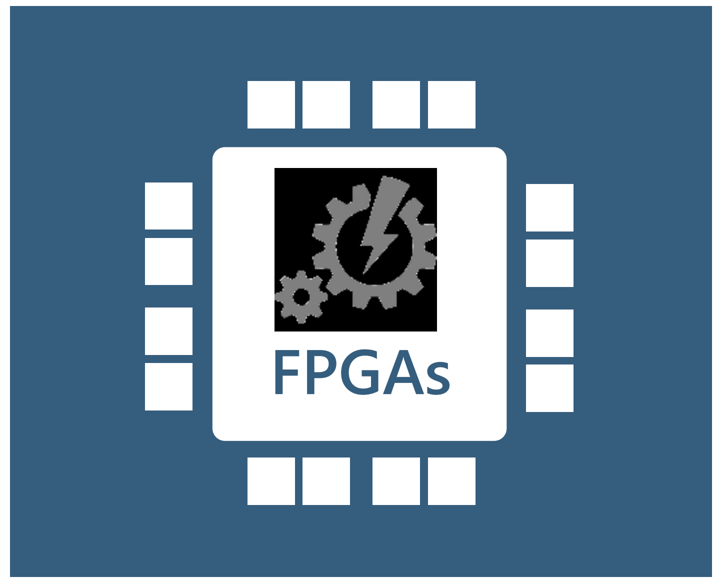
PROGRAMMING HARDWARE (FPGAS)

For more: https://youtu.be/iHg0mmIg0UU

- Say you want to program an "adder" function on an FPGA

```
module adder(
    input  wire [4:0] a,
    input  wire [4:0] b,
    output wire [4:0] y
);
    assign y = a + b;

endmodule
```

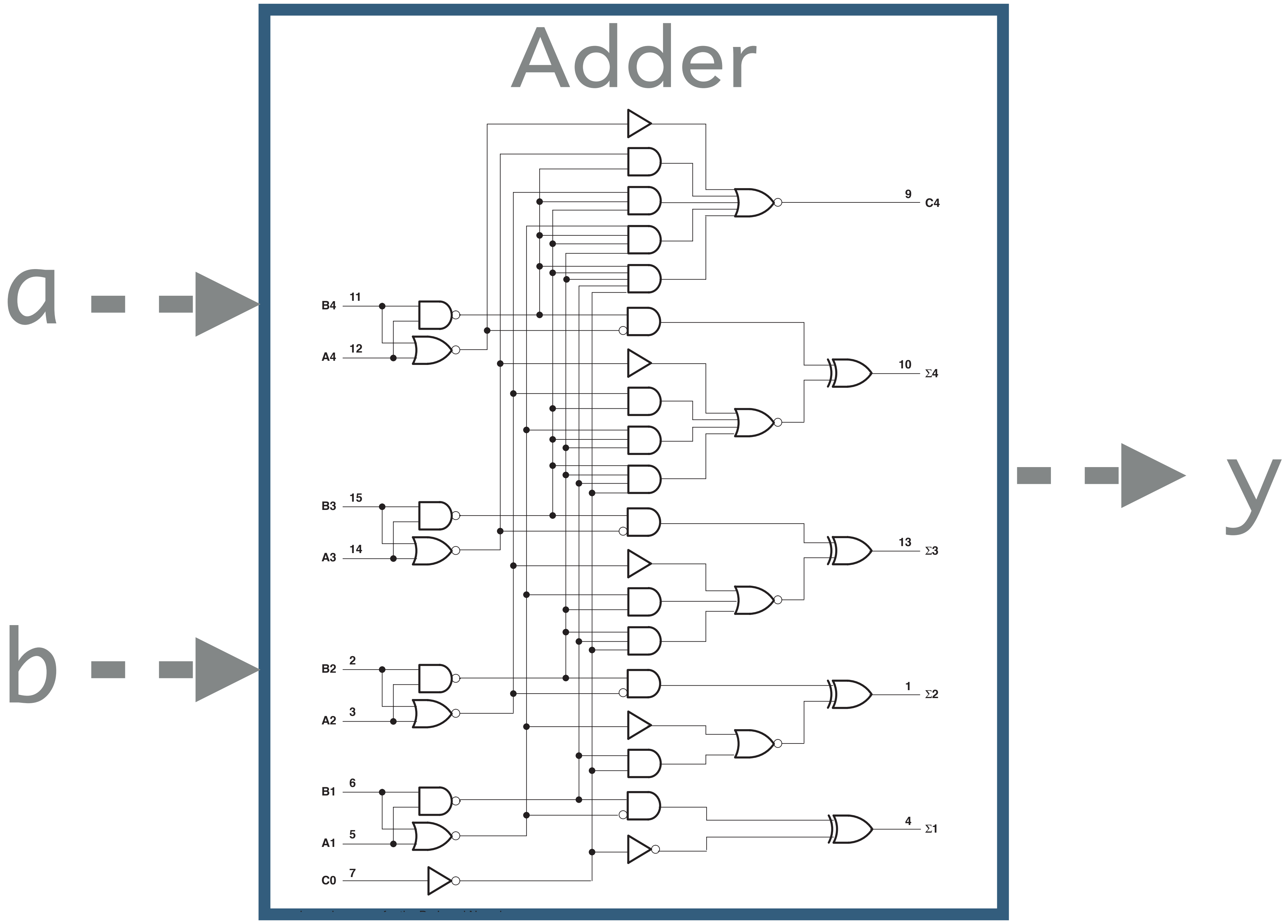


- Register transfer-level (RTL) code is "synthesized" into gates

- Say you want to program an "adder" function on an FPGA

```
module adder(
      input  wire [4:0] a,
      input  wire [4:0] b,
      output wire [4:0] y
);
      assign y = a + b;

endmodule
```

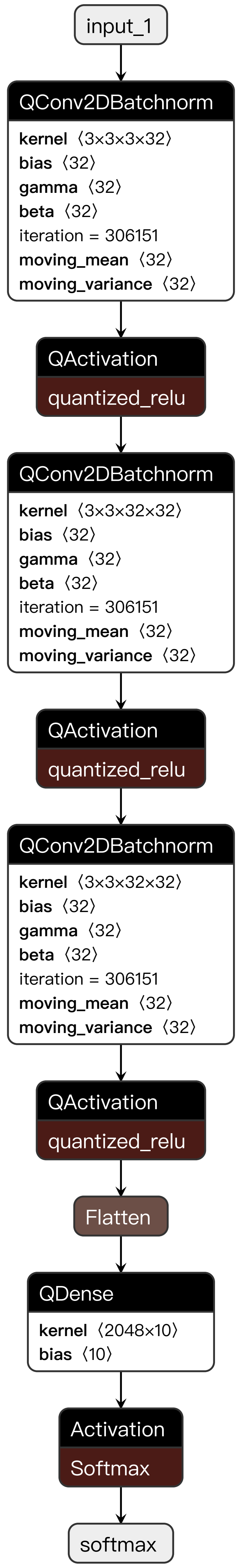


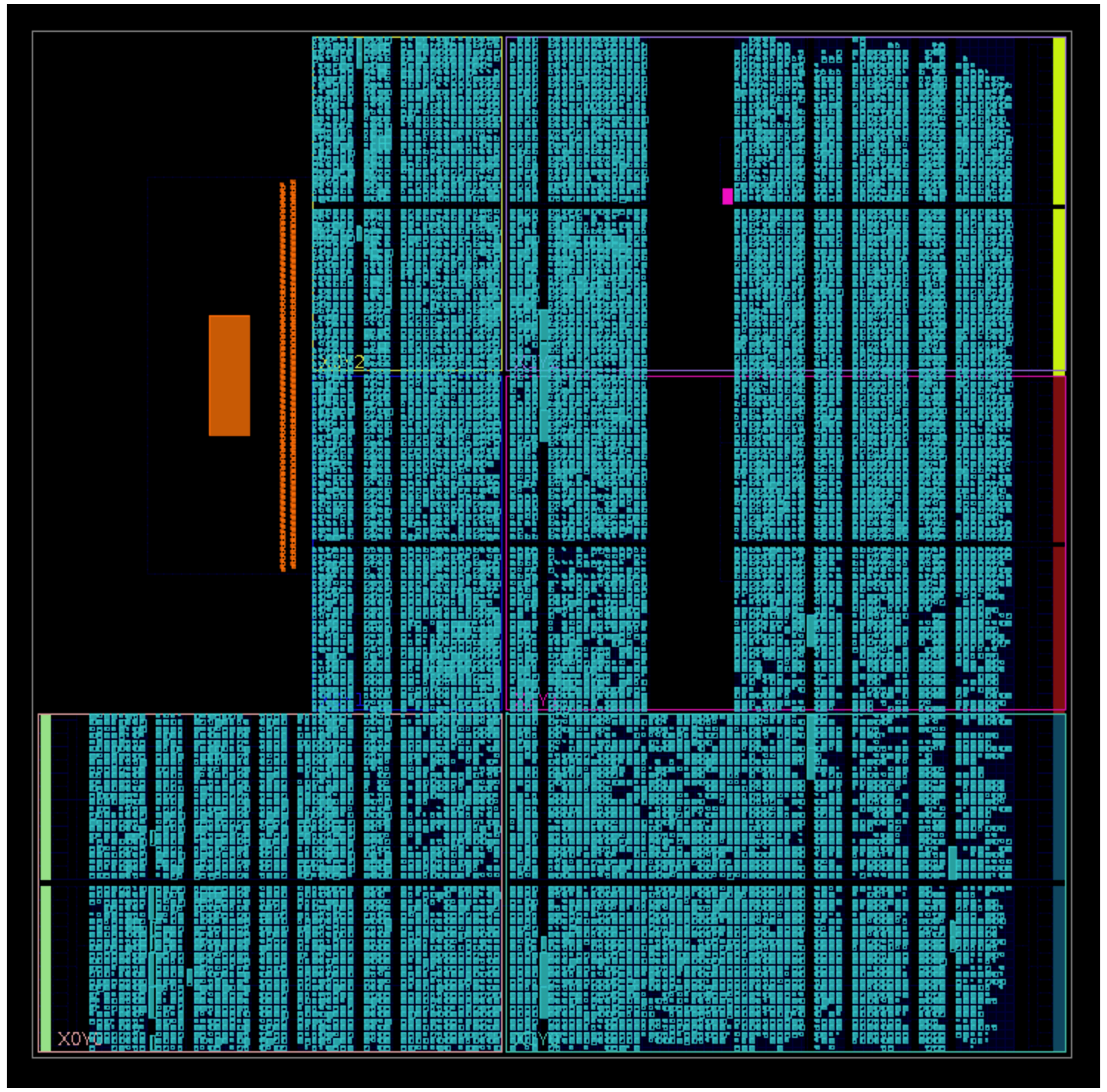
- Register transfer-level (RTL) code is "synthesized" into gates

Synthesis

PROGRAMMING HARDWARE (FPGAS)

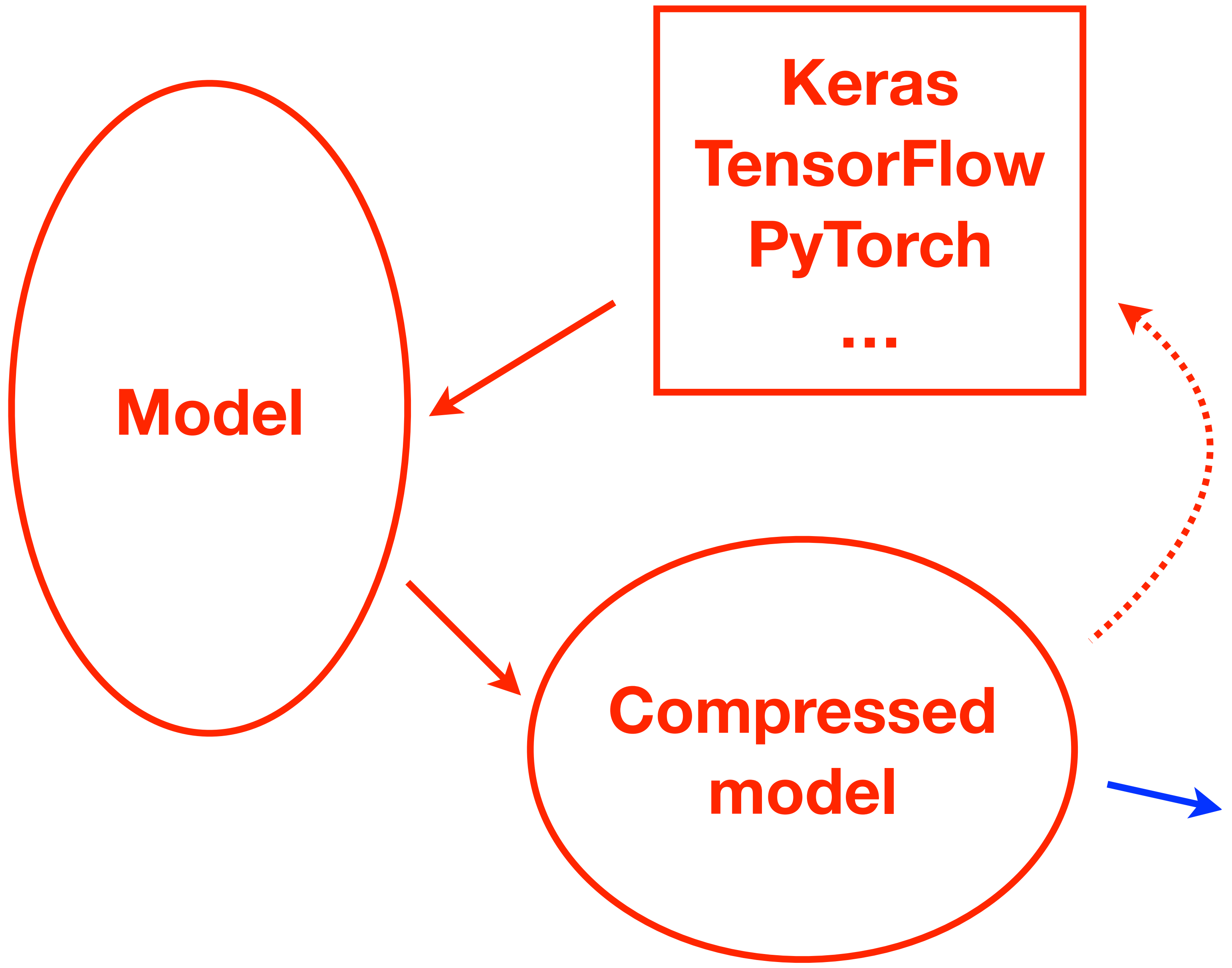
- What if instead we specify an AI model





High-Level Synthesis

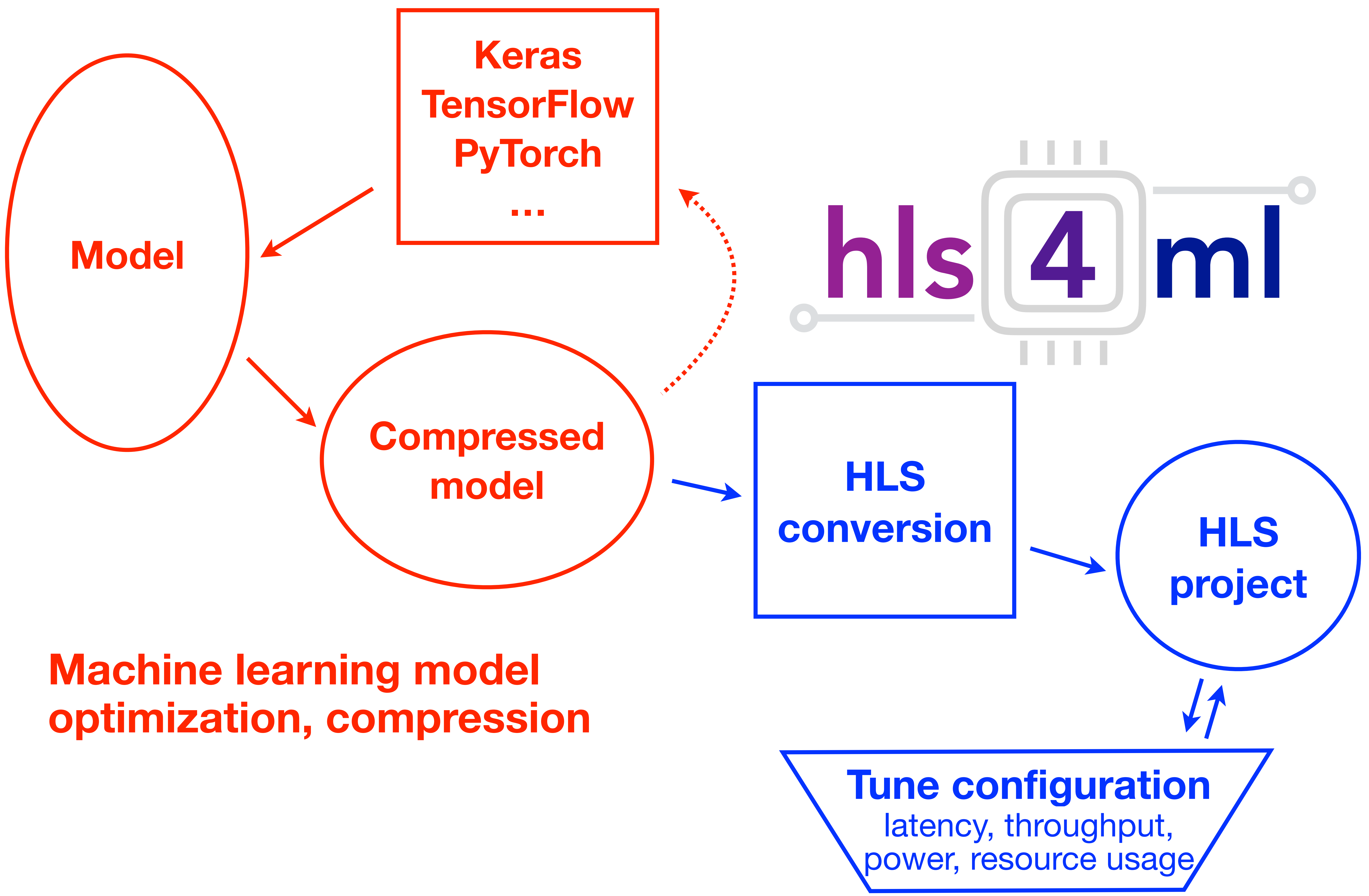
- hls4ml for scientists or ML experts to translate ML algorithms into RTL firmware



Machine learning model optimization, compression

DESIGN EXPLORATION WITH HLS4ML

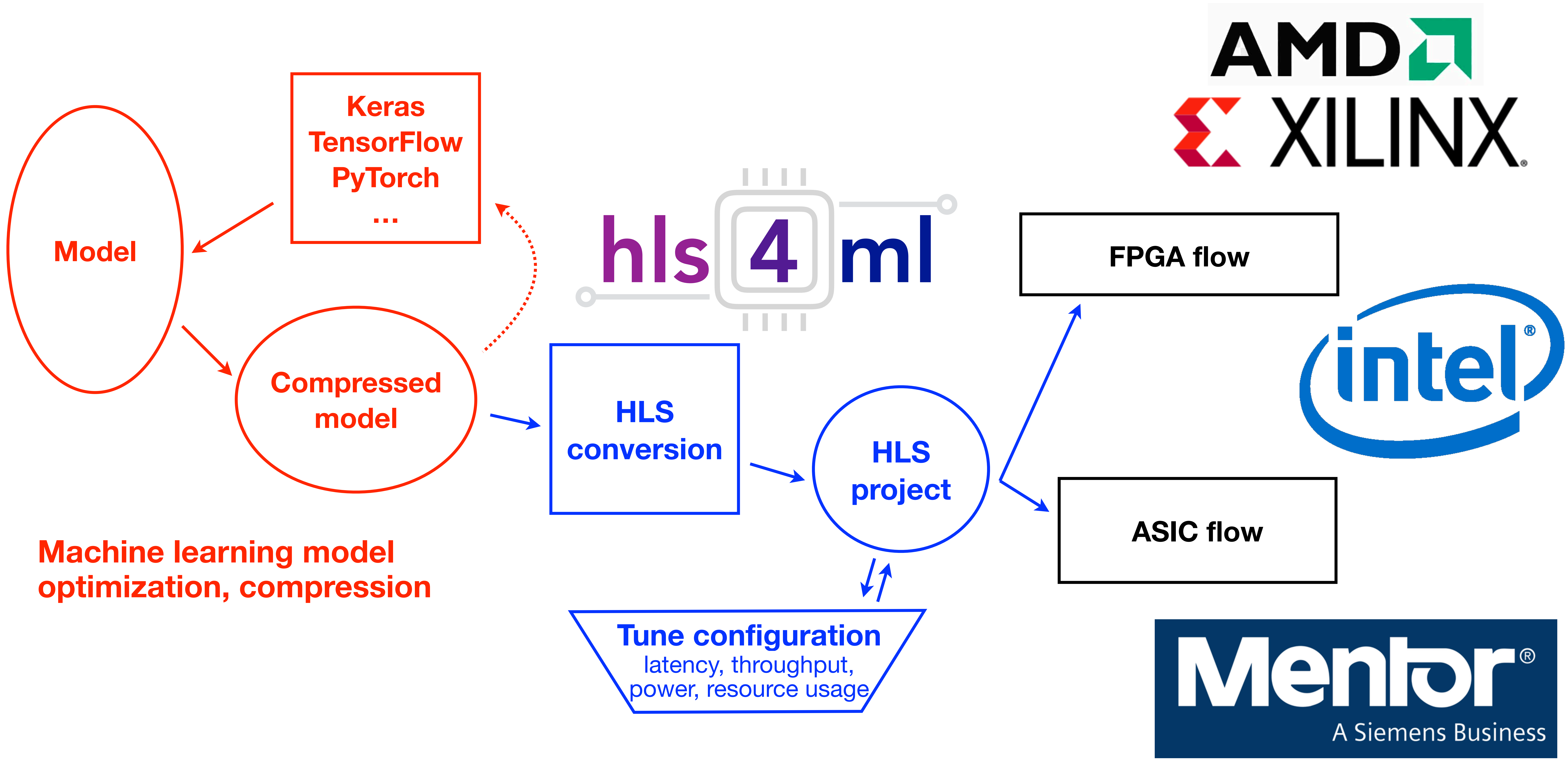
- hls4ml for scientists or ML experts to translate ML algorithms into RTL firmware



DESIGN EXPLORATION WITH HLS4ML

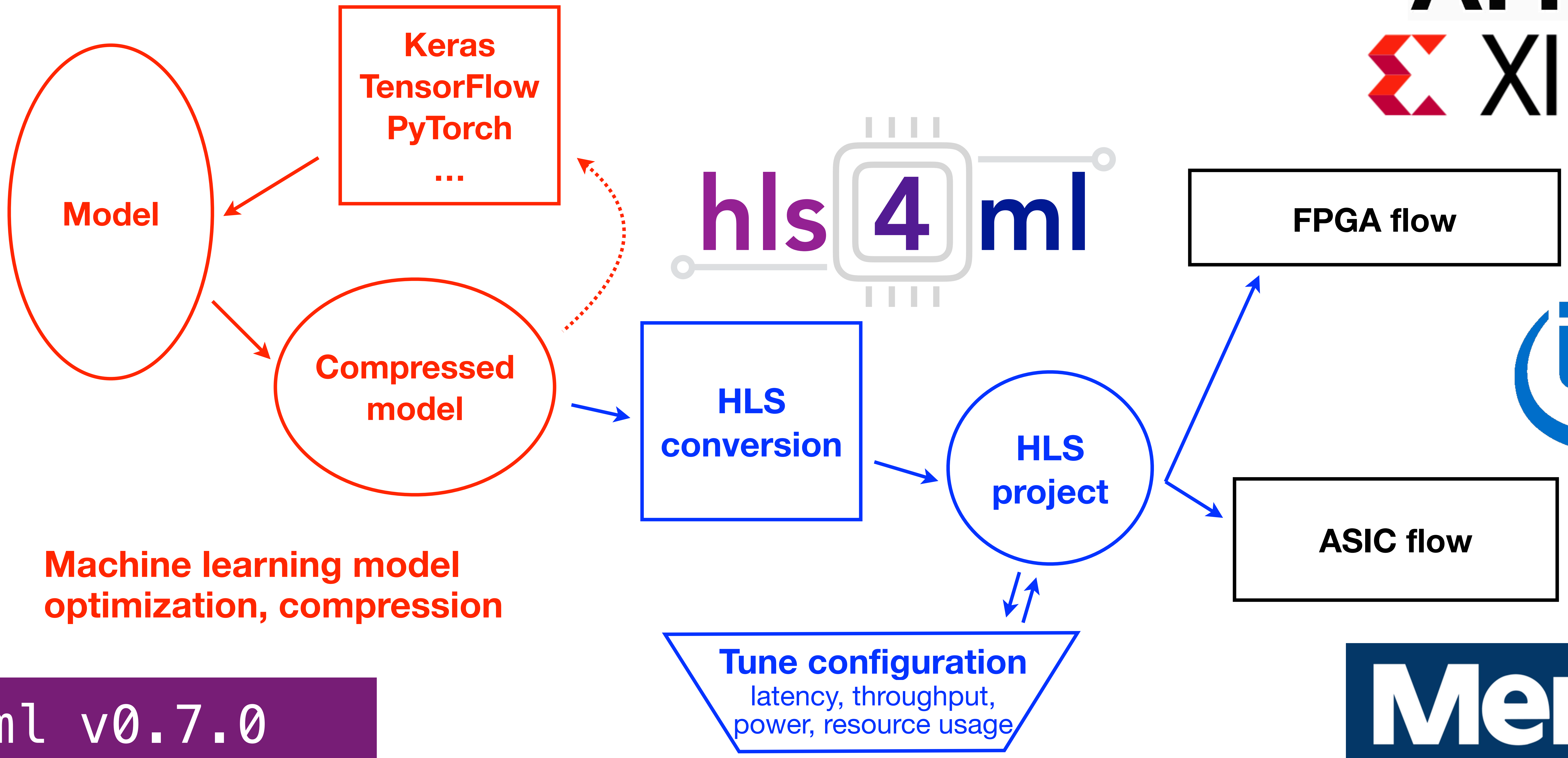
JINST 13, P07027 (2018)

- hls4ml for scientists or ML experts to translate ML algorithms into RTL firmware



DESIGN EXPLORATION WITH HLS4ML

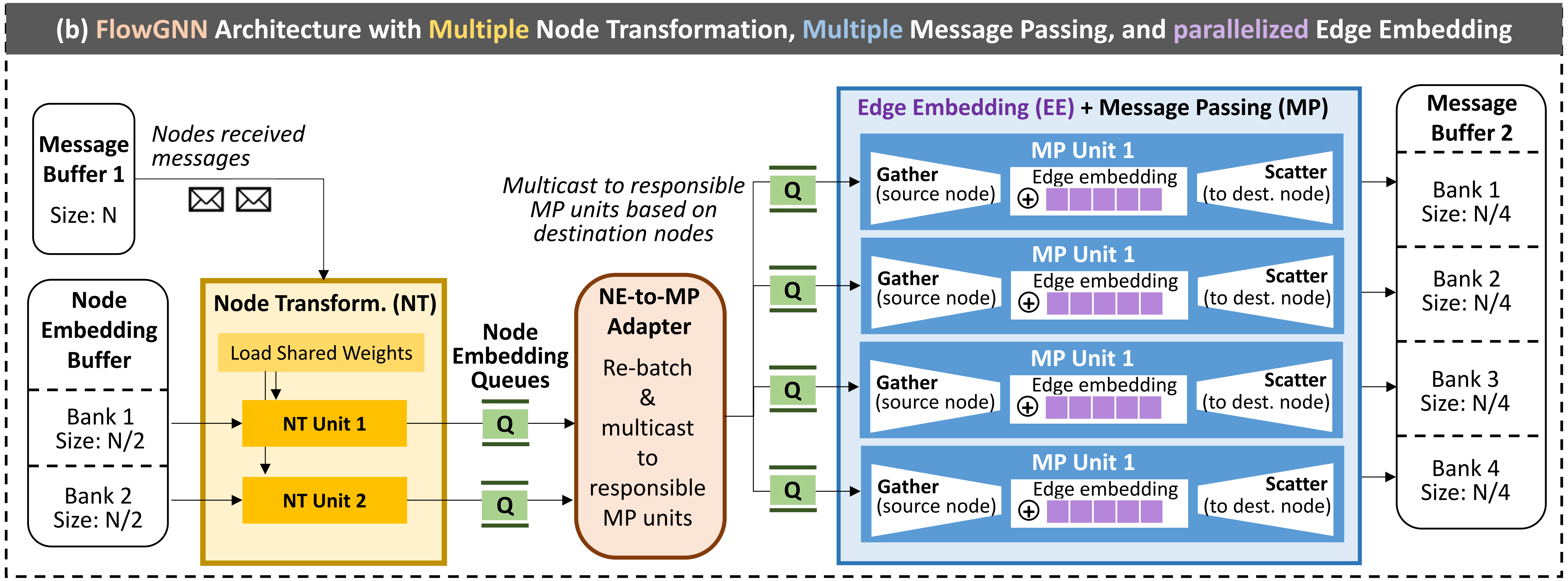
- hls4ml for scientists or ML experts to translate ML algorithms into RTL firmware



hls4ml v0.7.0
coming this week!

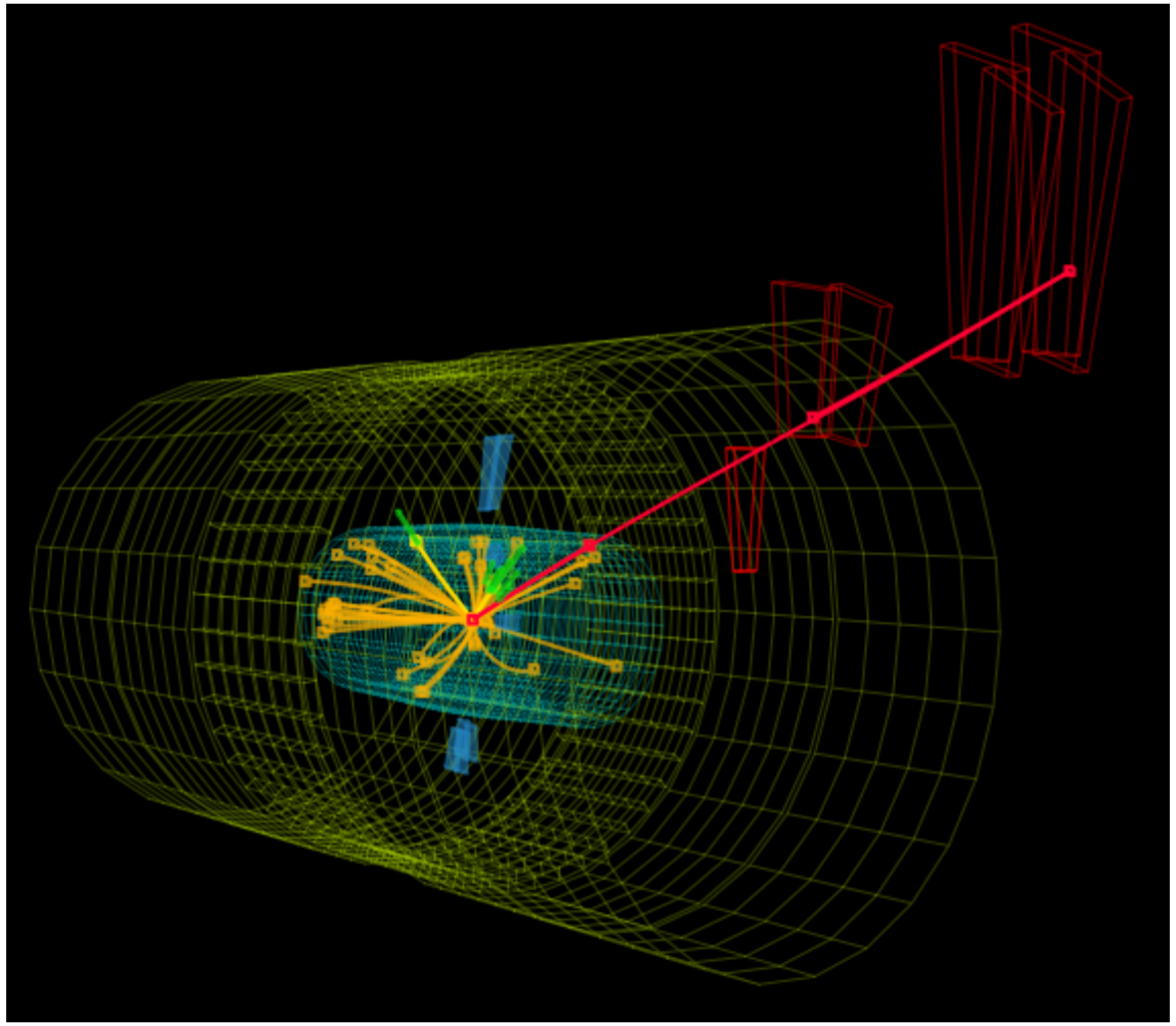
MANY TOOLS WITH DIFFERENT STRENGTHS

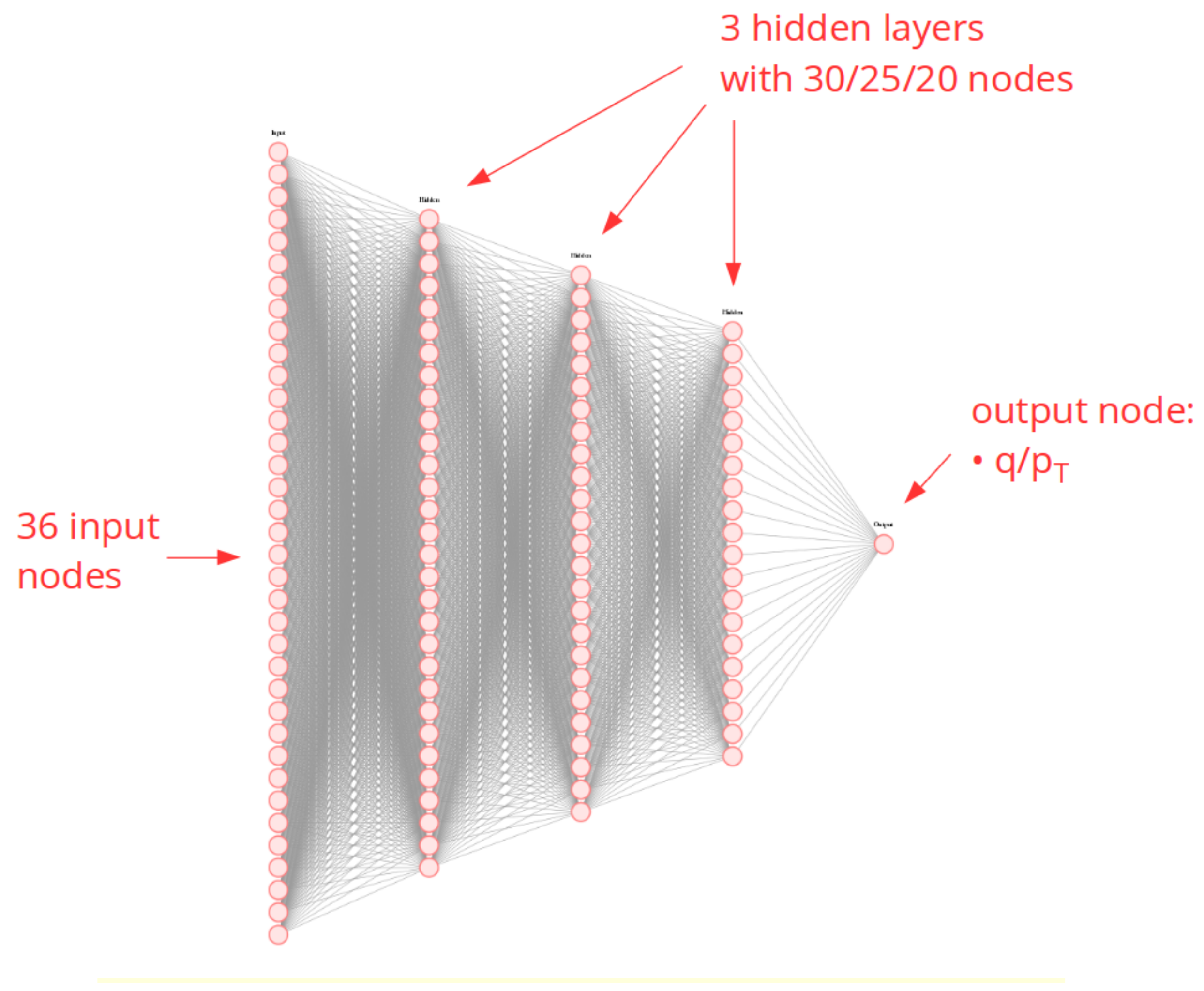
- FINN (NNs): https://finn.readthedocs.io/en/latest/
- Confier (BDTs): https://github.com/thesps/conifer
- fwXMachina (BDTs): http://fwx.pitt.edu/
- FlowGNN: https://github.com/sharc-lab/flowgnn



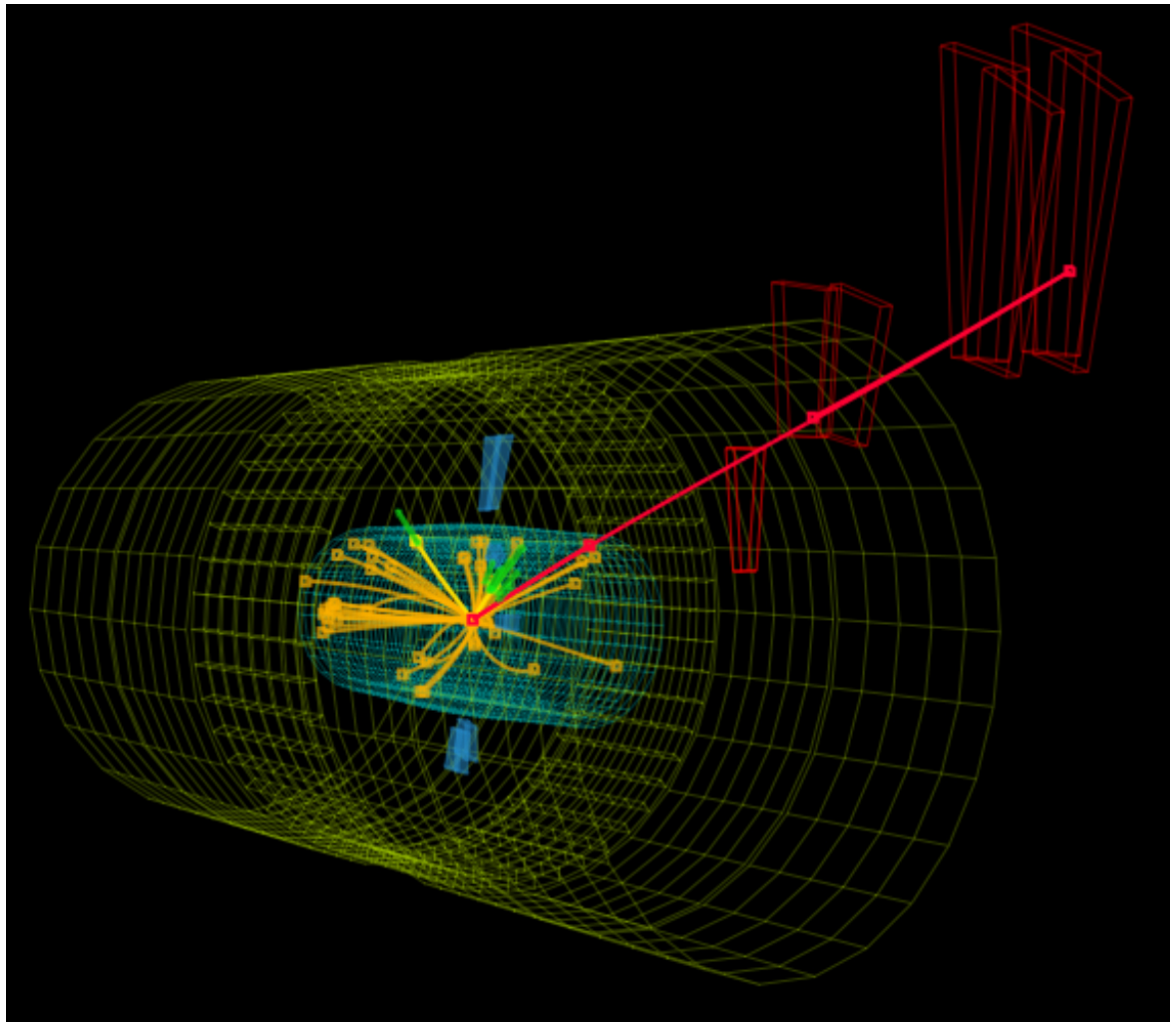

I. INTRO & MOTIVATION
II. COMPRESSION
III. HARDWARE
IV. APPLICATIONS

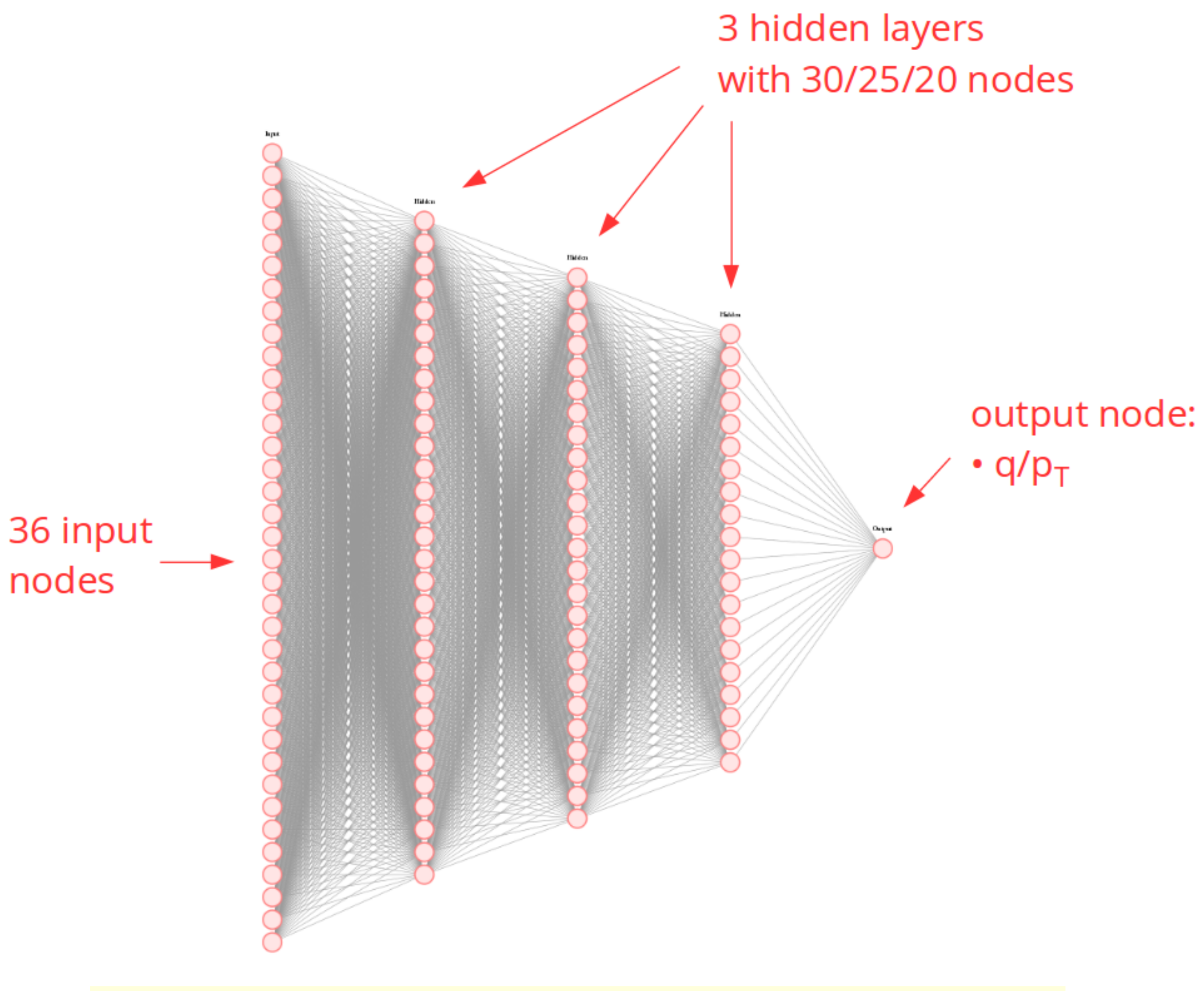
APPLICATION: MEASURE MUON PT AT 40 MHZ





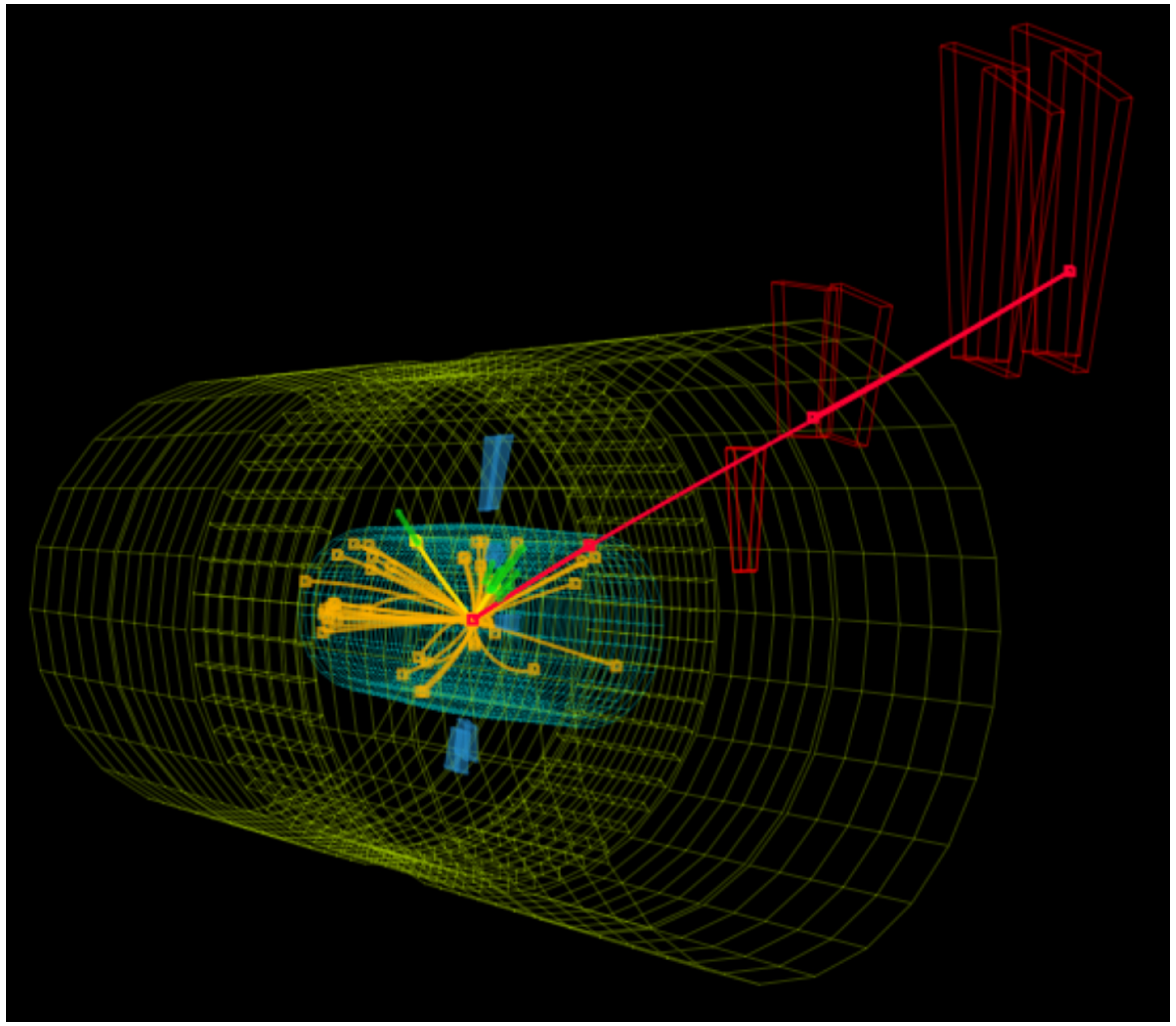
APPLICATION: MEASURE MUON PT AT 40 MHZ

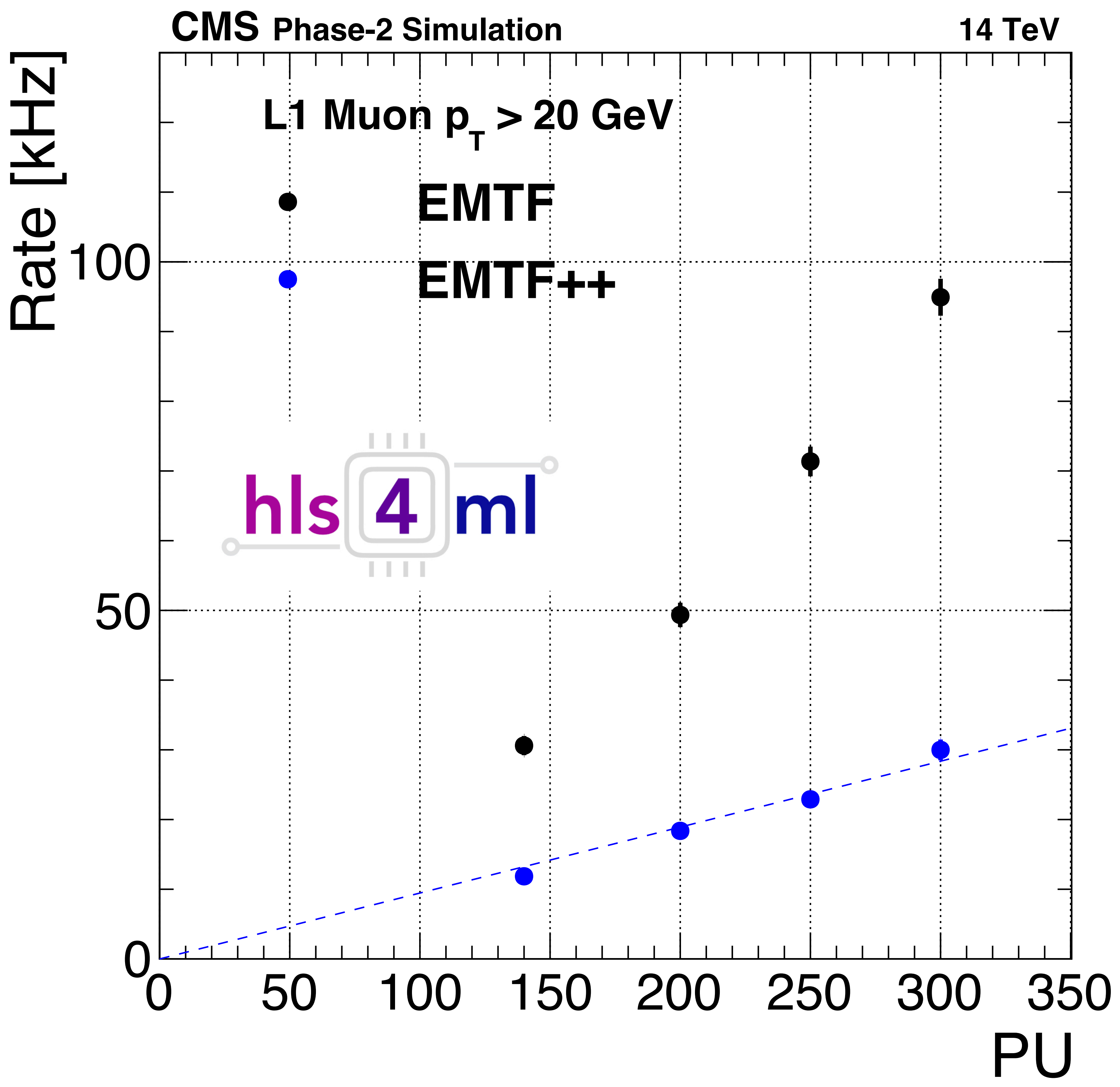


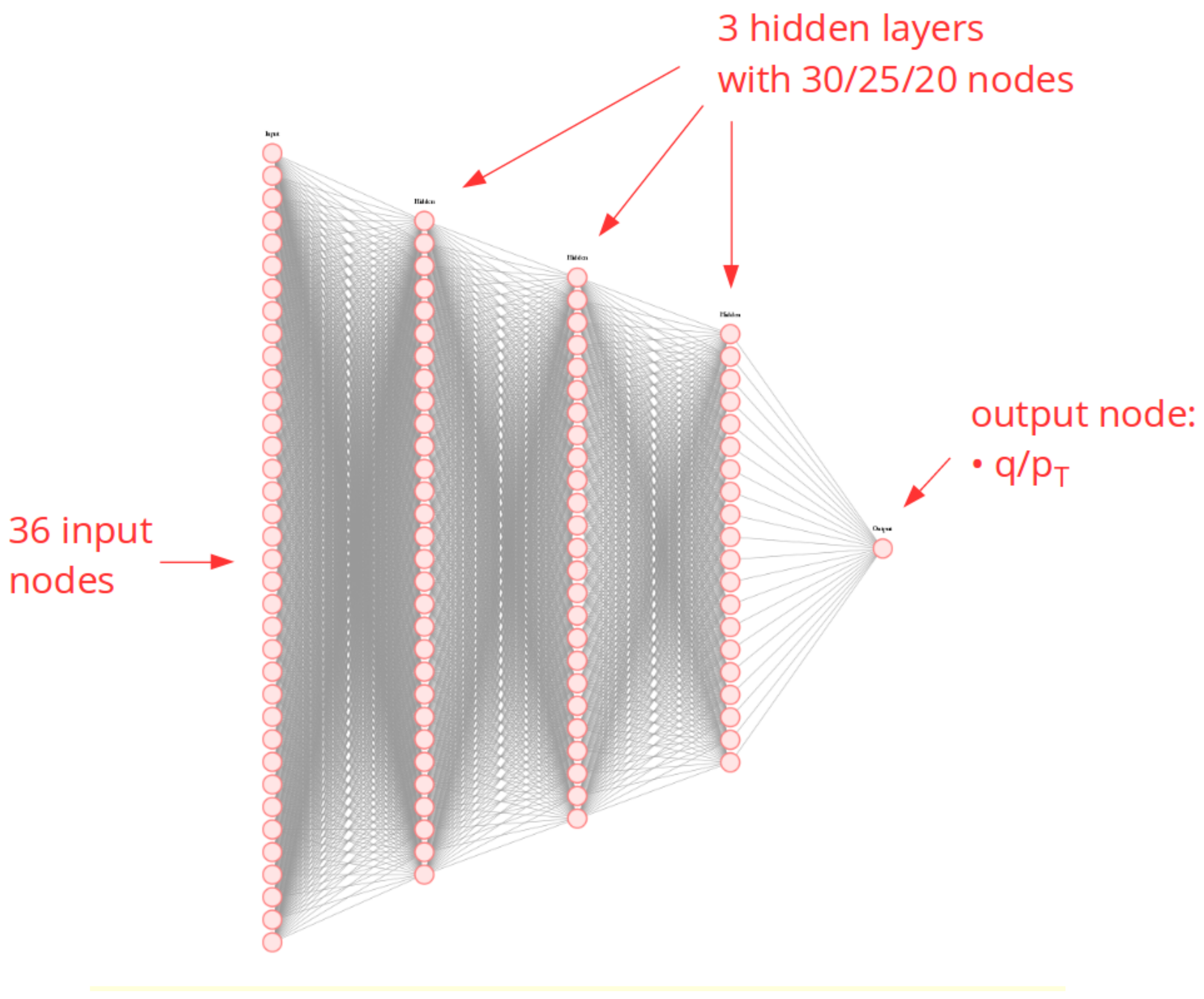


- NN measures muon momentum

APPLICATION: MEASURE MUON PT AT 40 MHZ

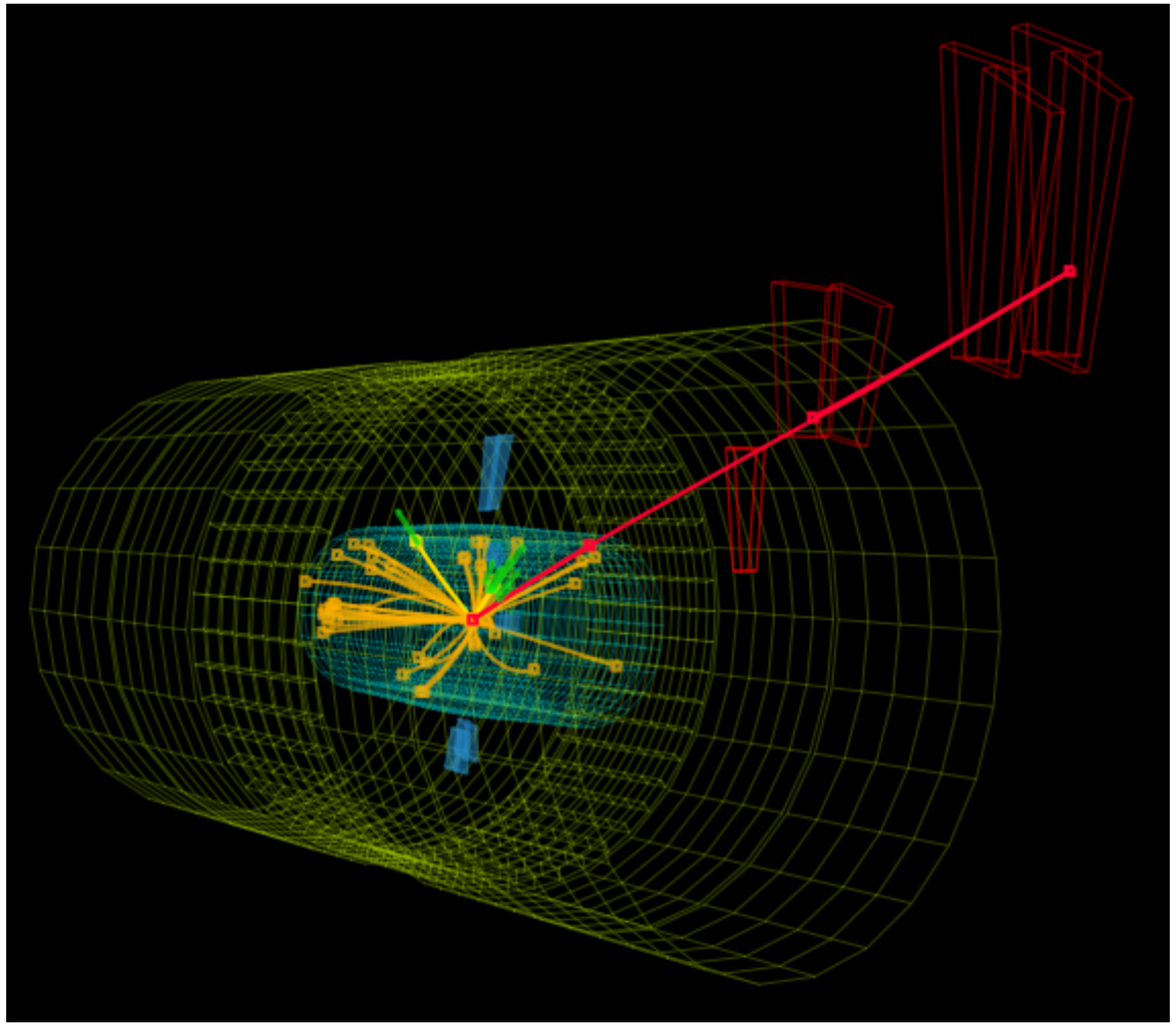


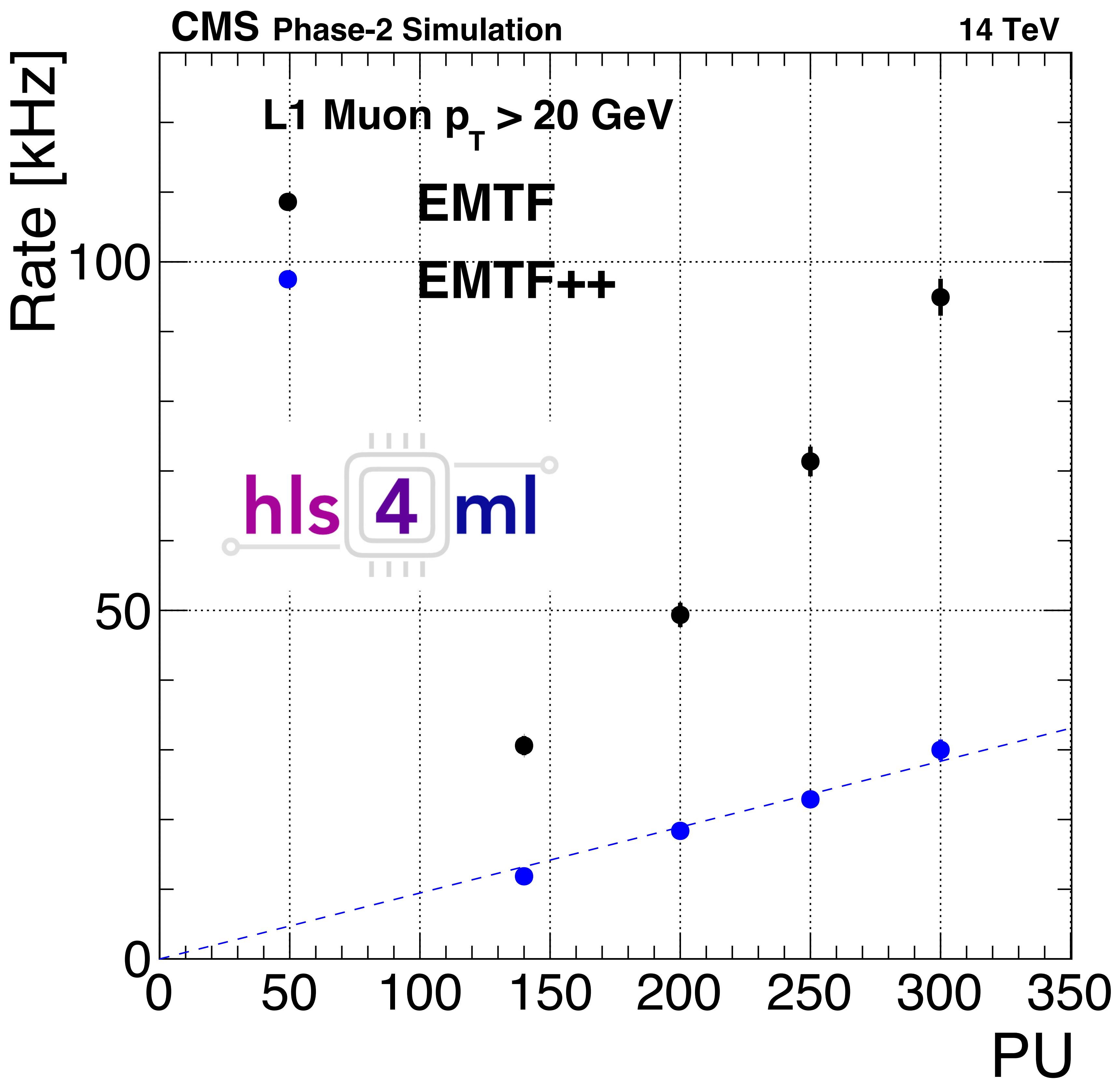


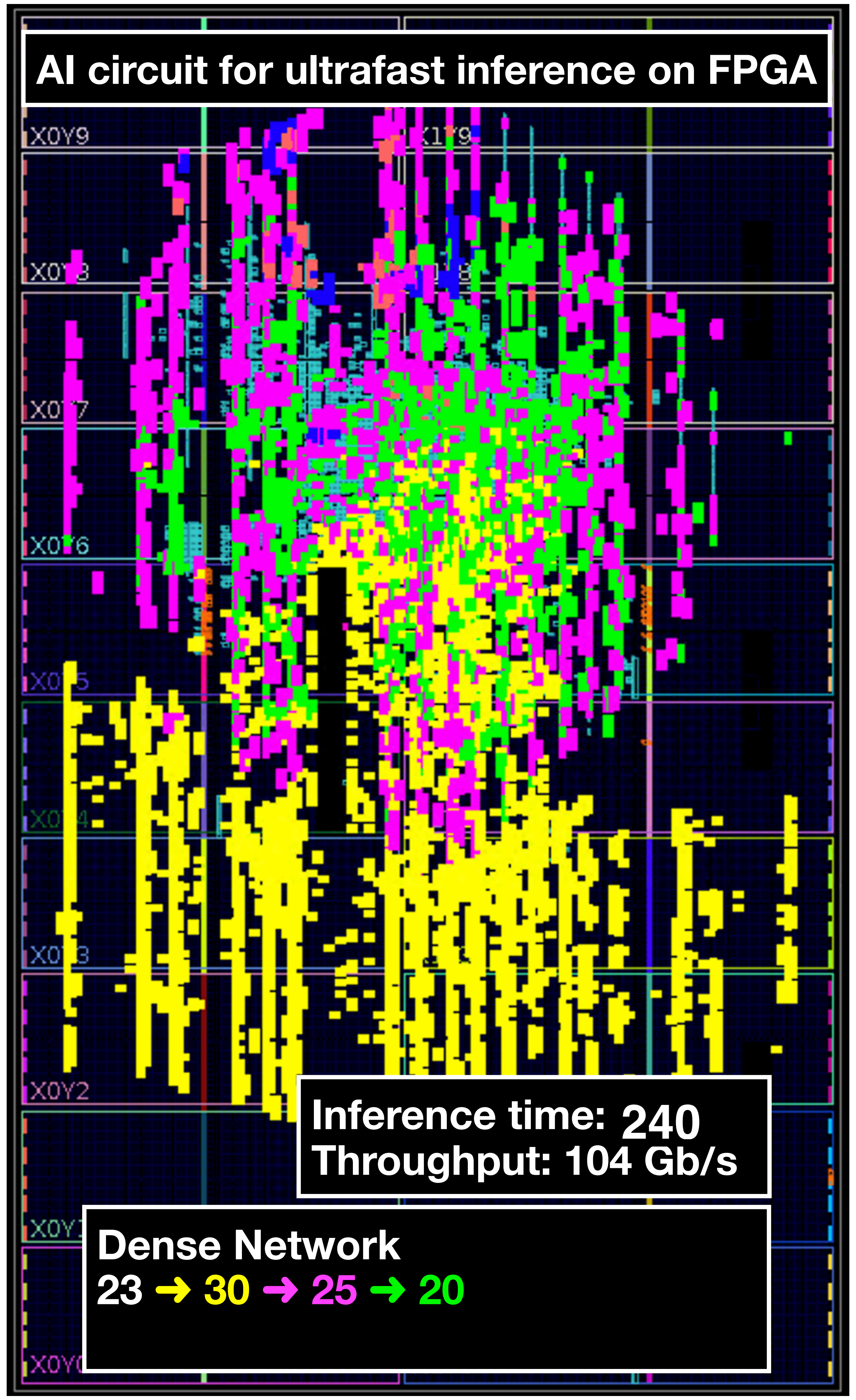


- NN measures muon momentum
  - 3× reduction in the trigger rate for NN!

APPLICATION: MEASURE MUON PT AT 40 MHZ

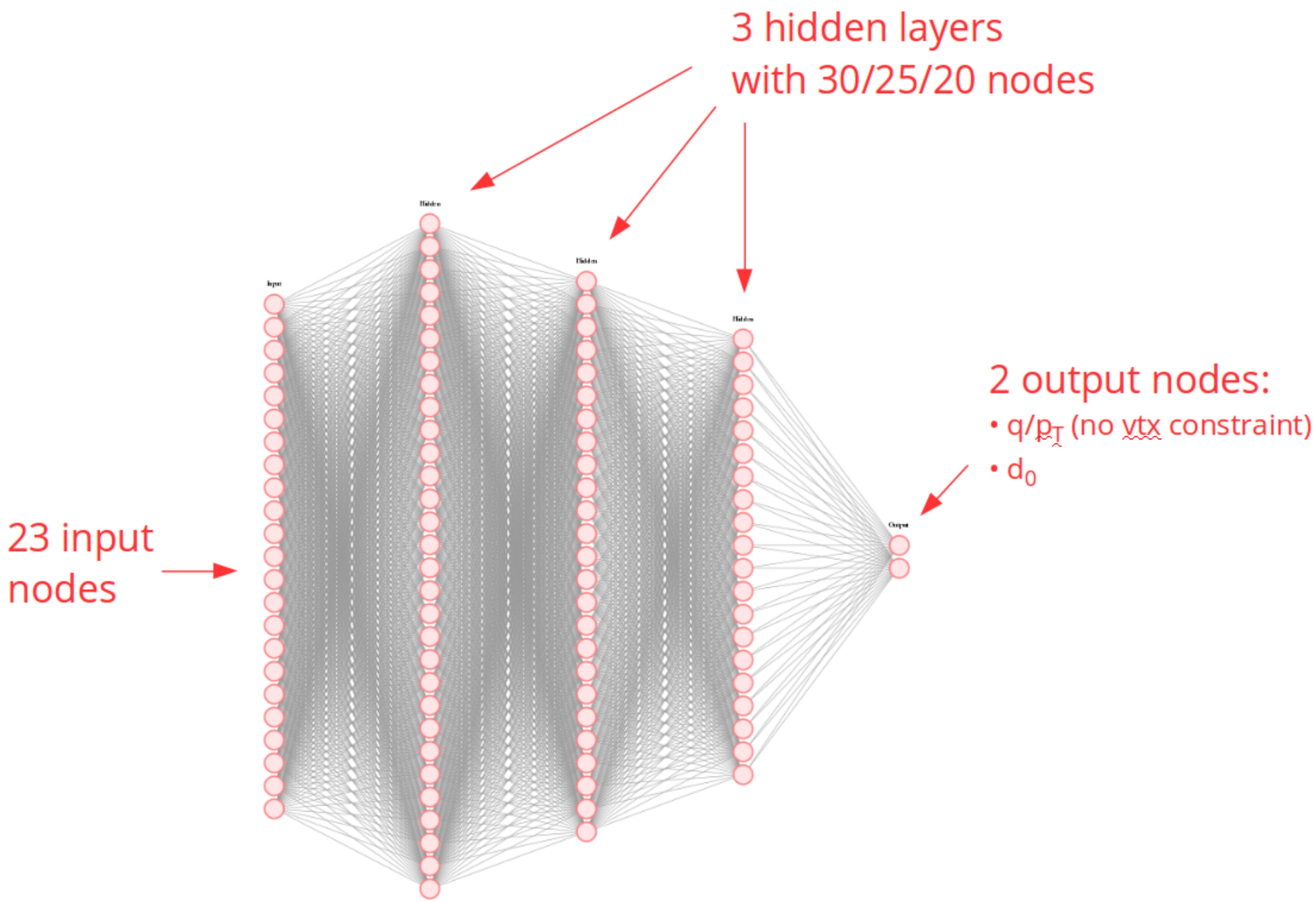






- NN measures muon momentum
  - 3× reduction in the trigger rate for NN!
- Fits within L1 trigger latency (240 ns!) and FPGA resource requirements (less then 30%)

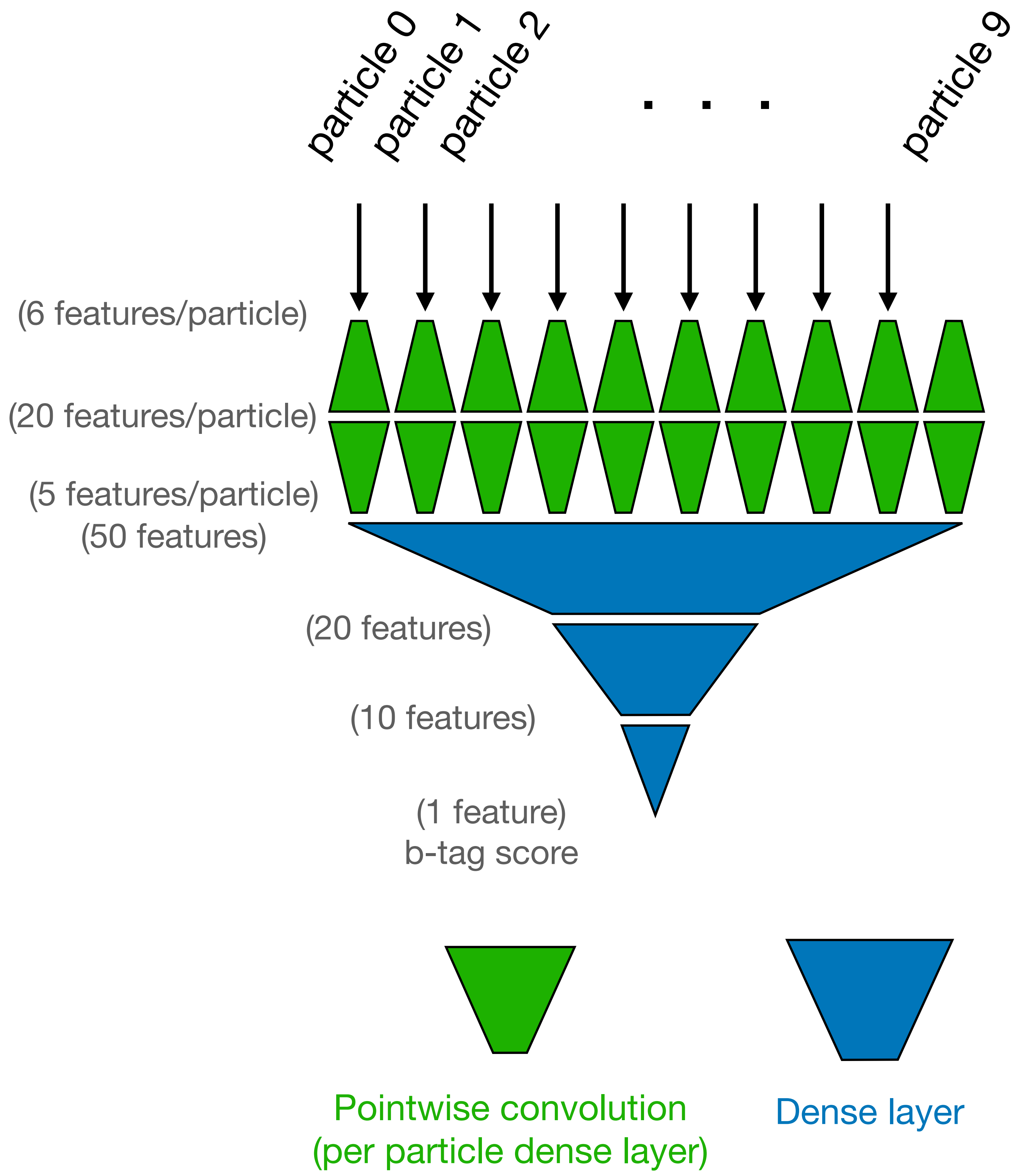
EXTENSION: MEASURE MUON DISPLACEMENT

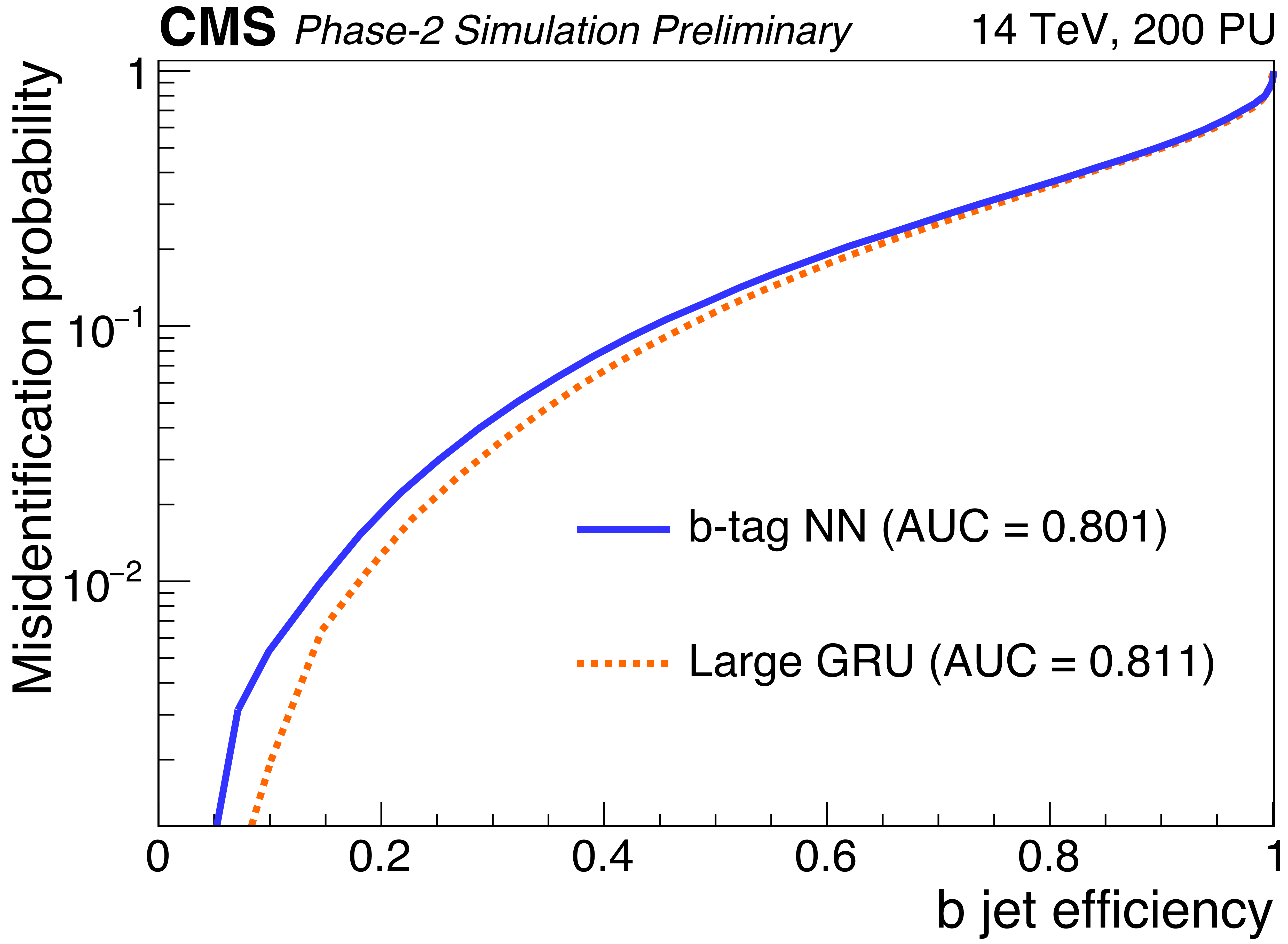


- Extends idea to measure muon displacement as well as p_T
- *Stay tuned for Run 3 results*

APPLICATION: B-TAGGING @ L1

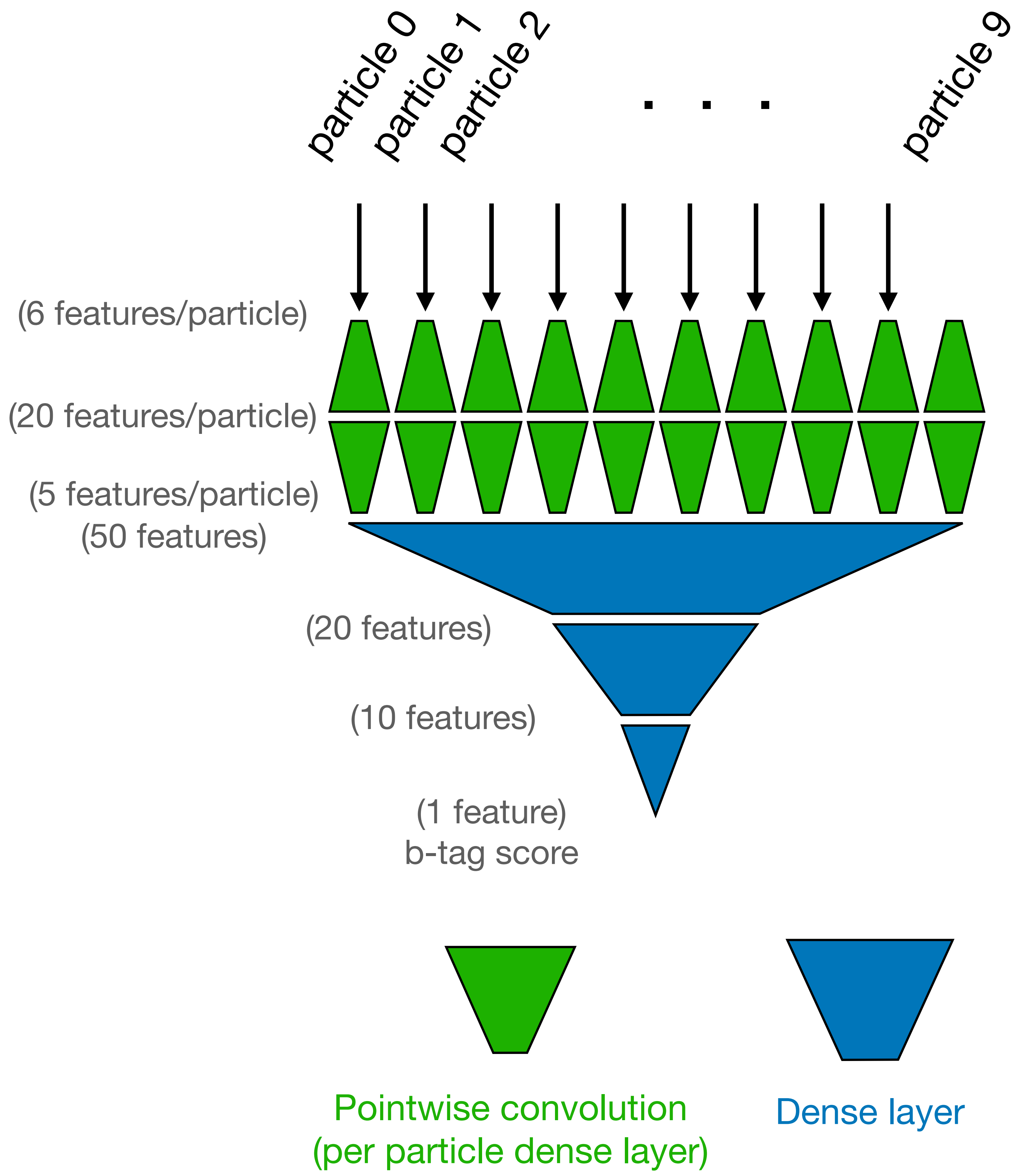
- Upgraded HL-LHC level-1 track trigger information enables b-tagging with a ***neural network*** to improve the $HH \to 4b$ search
  - Input features for 10 particles within each jet: particle type, momentum, and vertex information

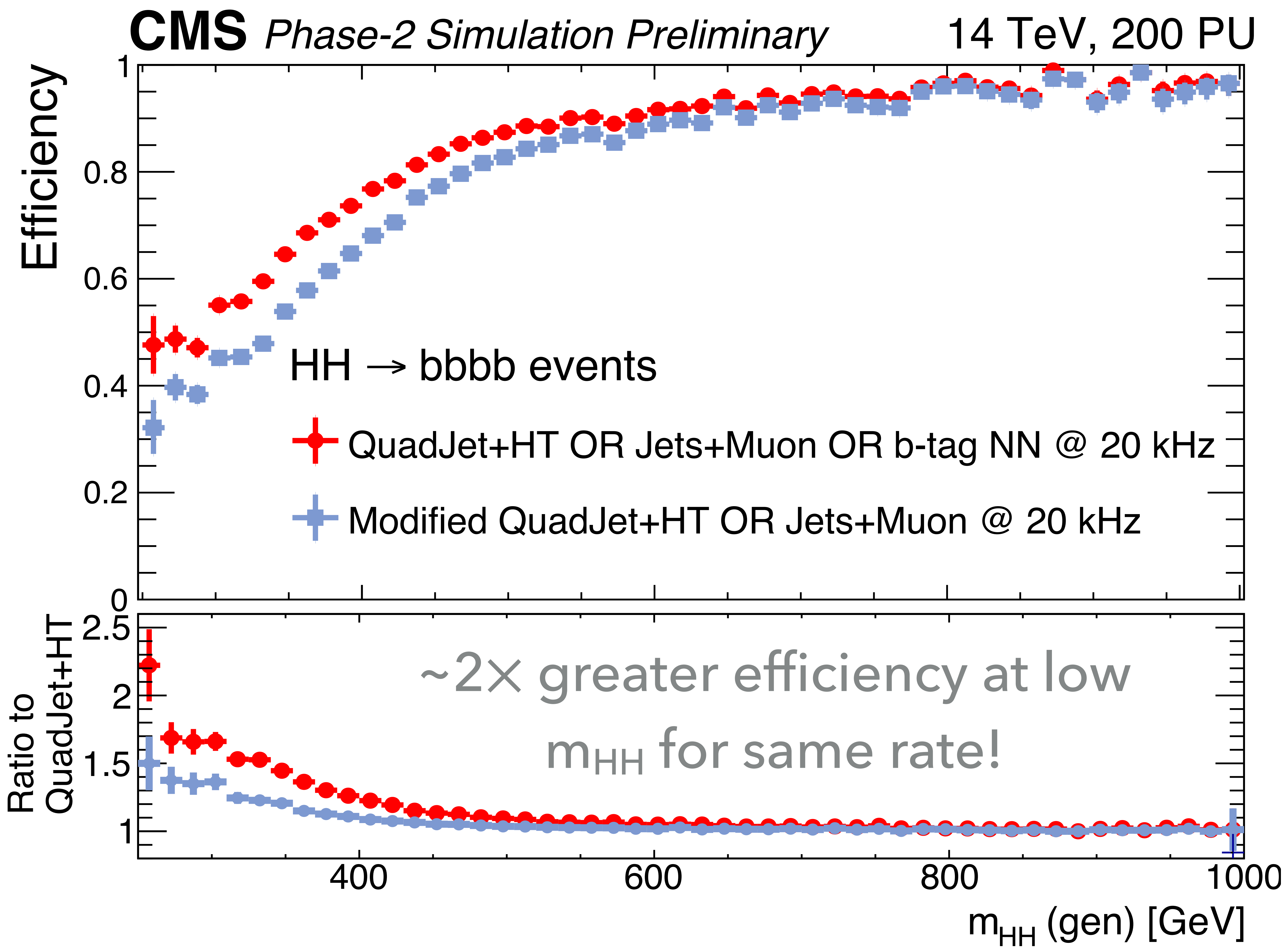




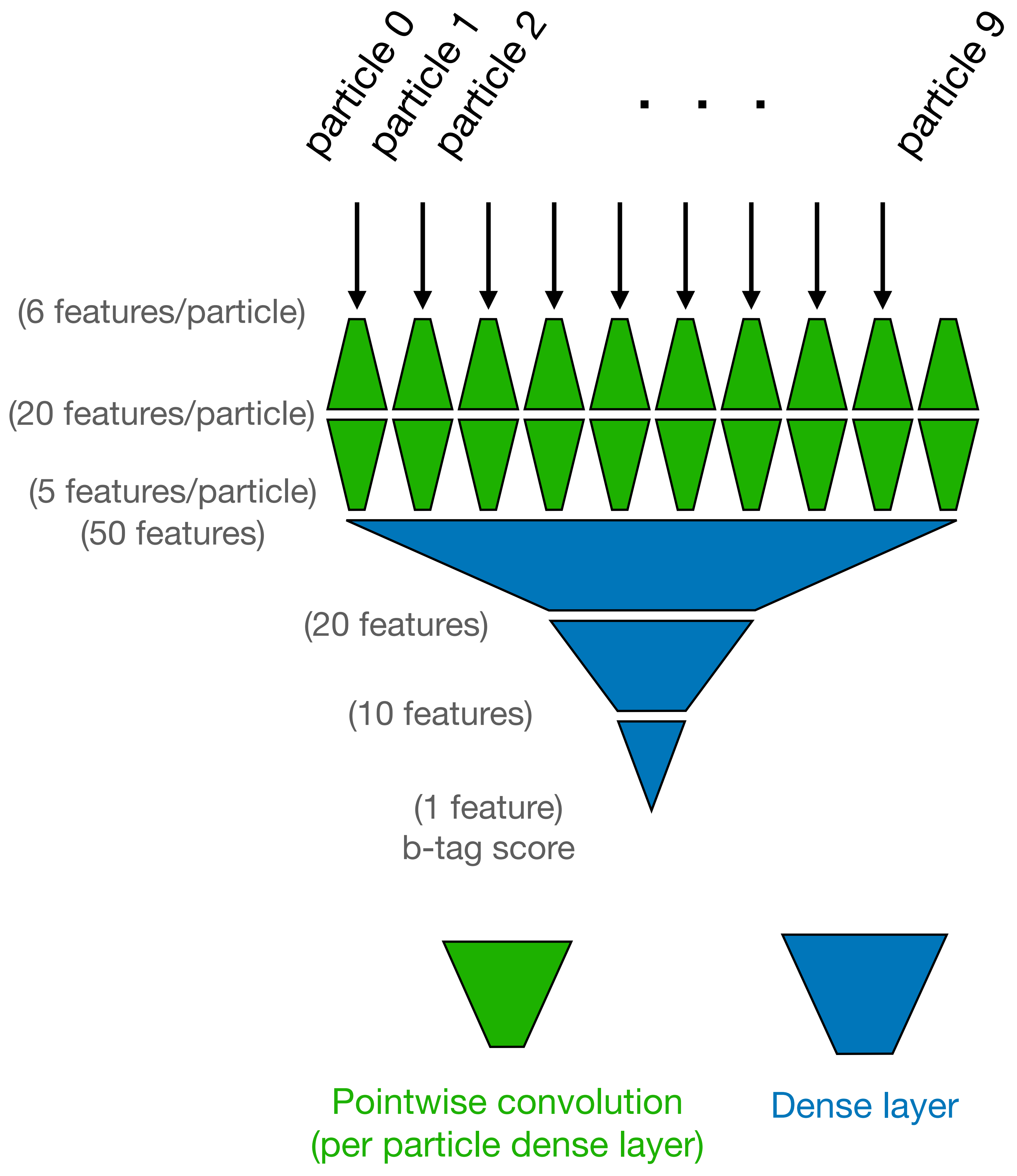
APPLICATION: B-TAGGING @ L1

- Upgraded HL-LHC level-1 track trigger information enables b-tagging with a ***neural network*** to improve the $HH \to 4b$ search
  - Input features for 10 particles within each jet: particle type, momentum, and vertex information



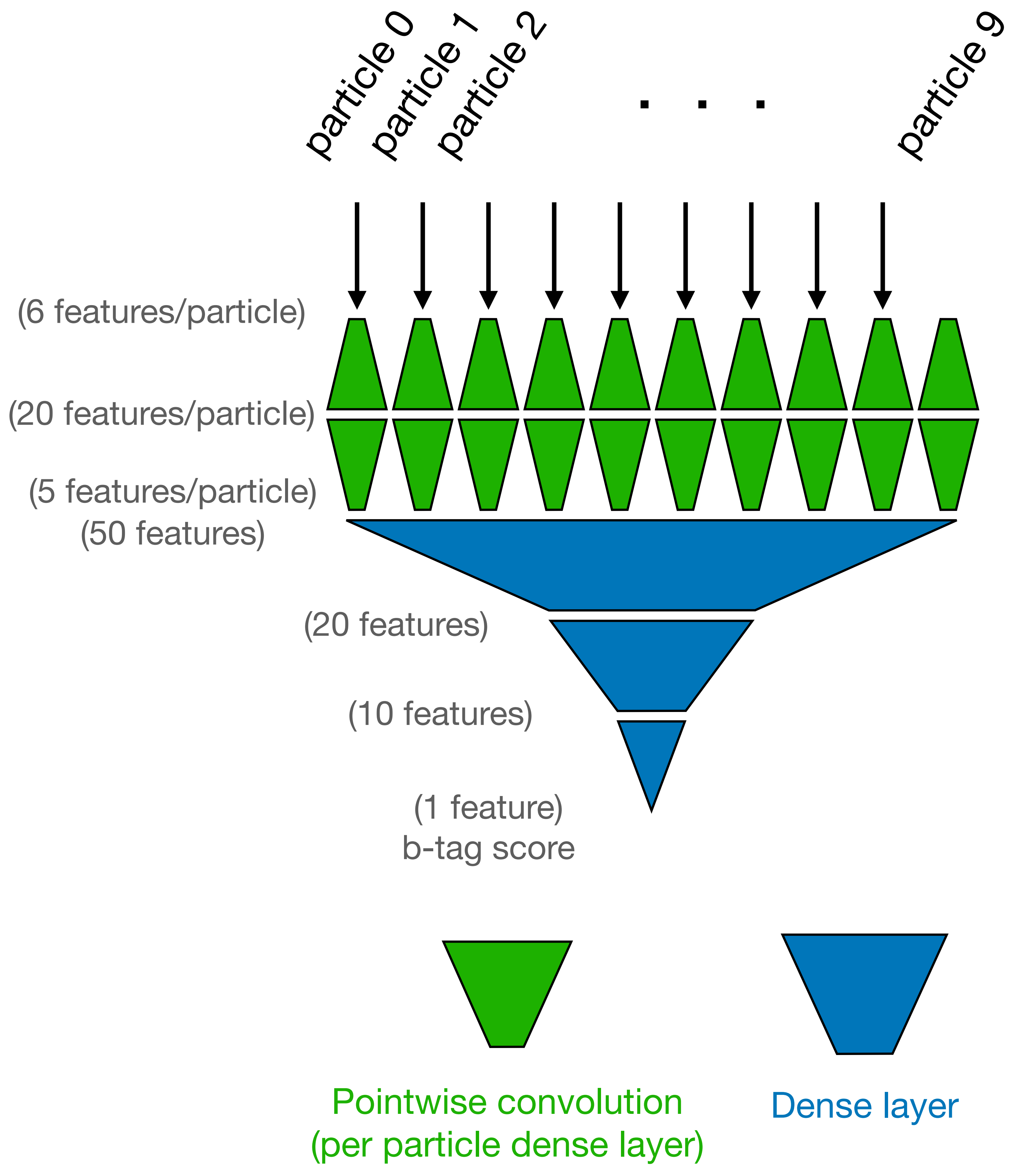


IMPLEMENTATION: B-TAGGING @ L1



IMPLEMENTATION: B-TAGGING @ L1

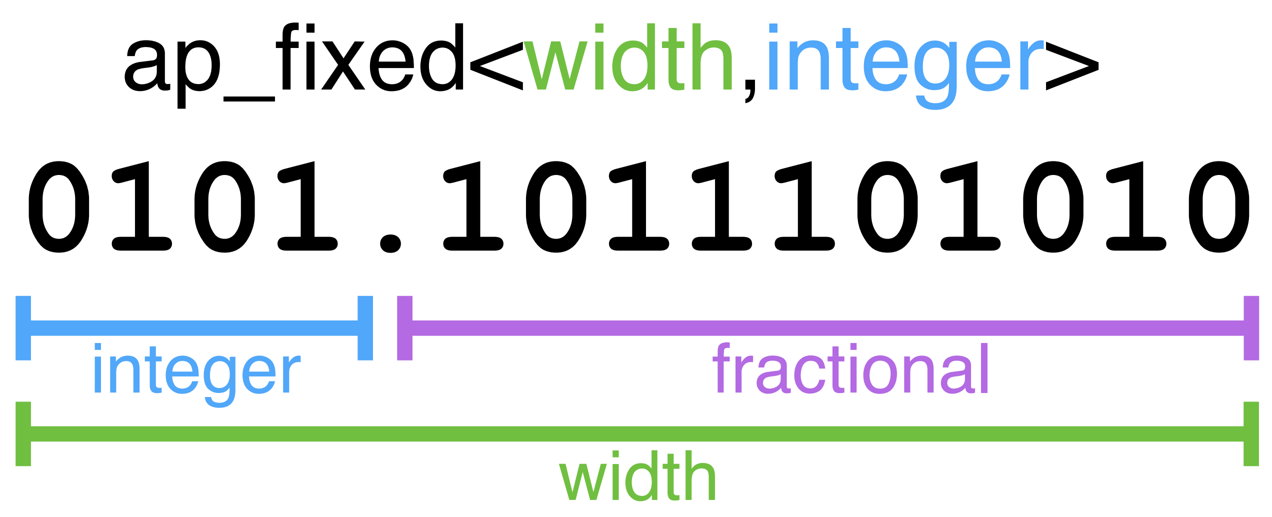
- But does it fit and meet timing?

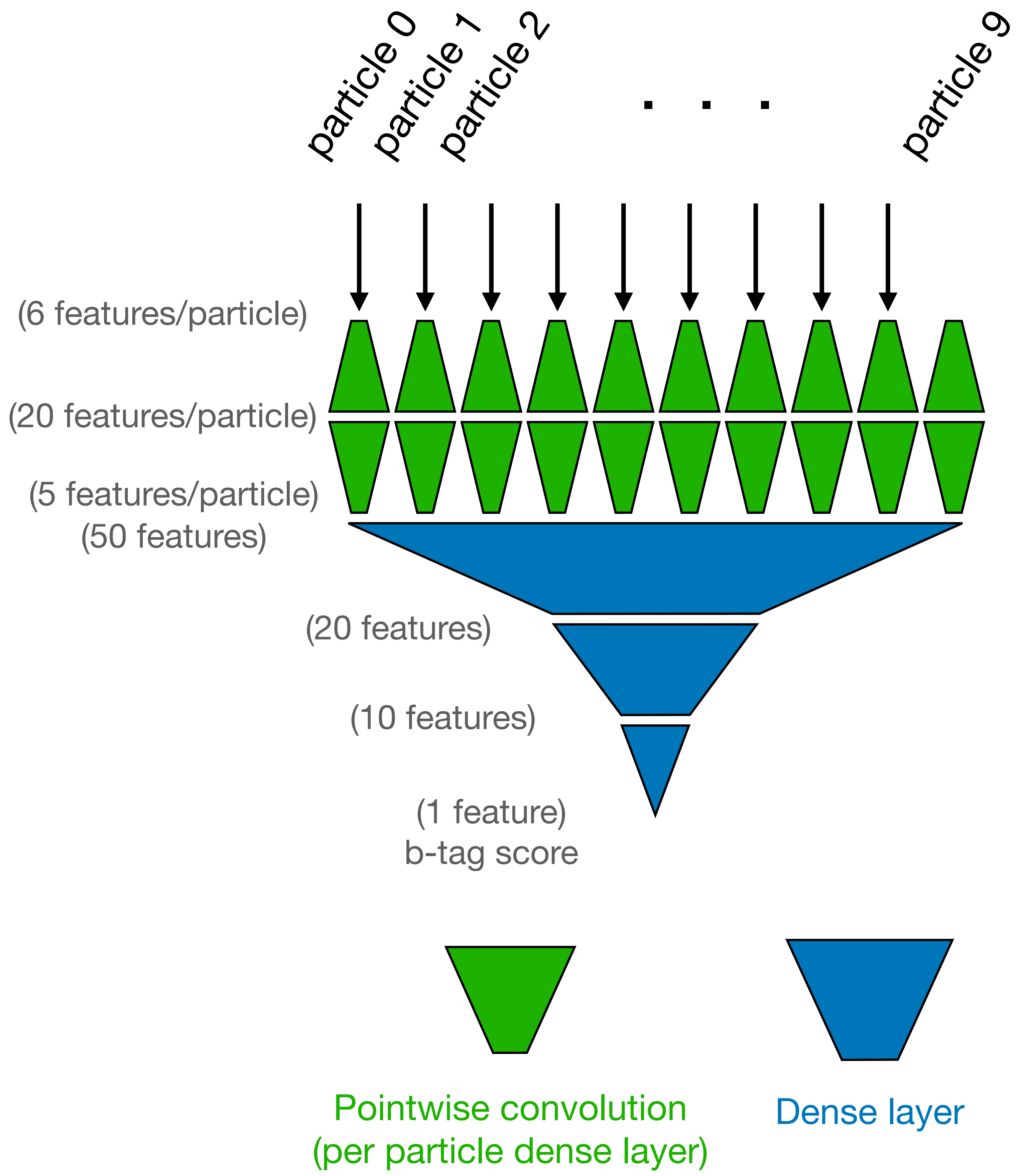


IMPLEMENTATION: B-TAGGING @ L1

CMS-DP-2022-021

- But does it fit and meet timing?
- After ***quantization***, can implement NN with 9 bits





IMPLEMENTATION: B-TAGGING @ L1

CMS-DP-2022-021

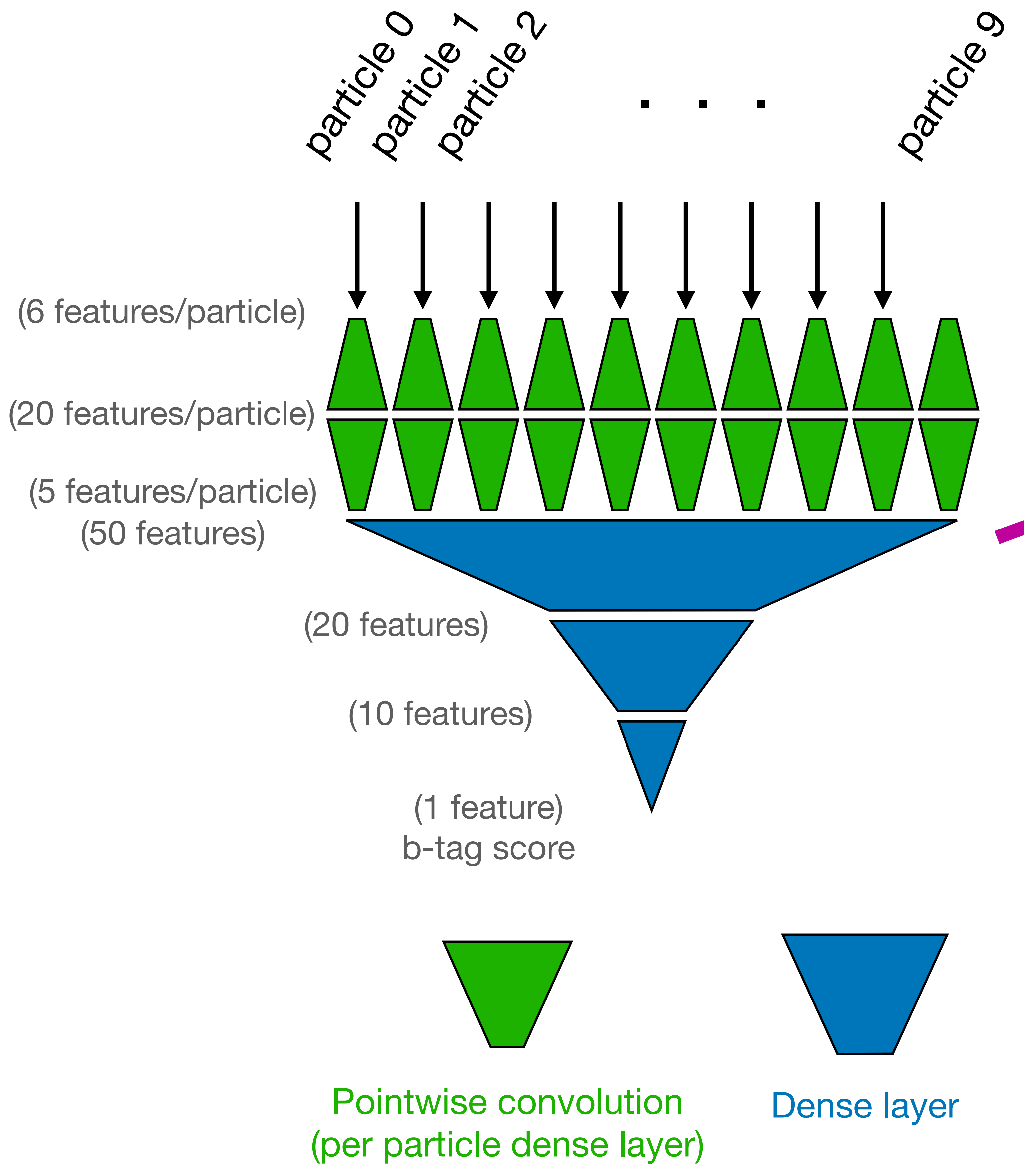
- But does it fit and meet timing?
- After ***quantization***, can implement NN with 9 bits
- Latency of 60 ns, II of 5 ns per jet, and <12% of FPGA

ap_fixed<width,integer>

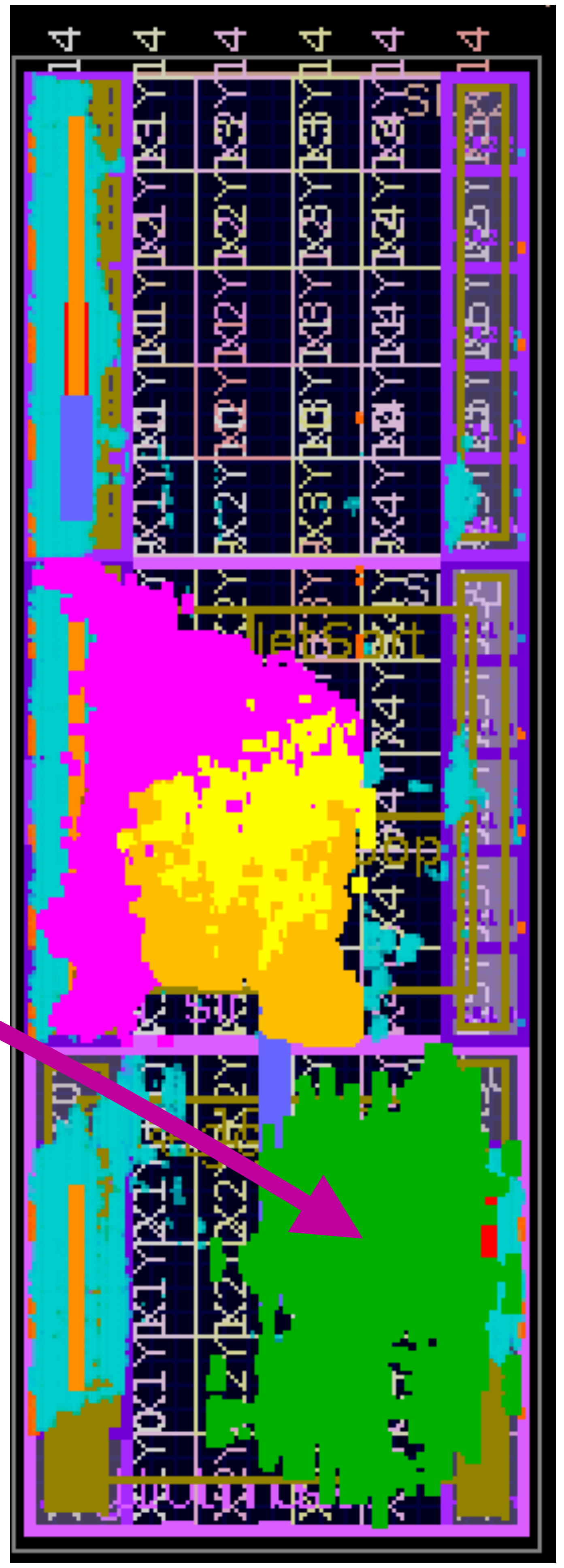
0101.1011101010

integer | fractional

width



hls4ml



Deregionizer

Jet loop

Jet sort

b-tag NN

APPLICATION: ANOMALY DETECTION AT 40 MHZ

[Nat. Mach. Intell. 4, 154 (2022)]
Data challenge: [mpp-hep.github.io/ADC2021]

APPLICATION: ANOMALY DETECTION AT 40 MHZ

Nat. Mach. Intell. 4, 154 (2022)
Data challenge: mpp-hep.github.io/ADC2021

- Challenge: if new physics has an unexpected signature that doesn't align with existing triggers, precious BSM events may be discarded at trigger level

APPLICATION: ANOMALY DETECTION AT 40 MHZ

Nat. Mach. Intell. 4, 154 (2022)

Data challenge: mpp-hep.github.io/ADC2021

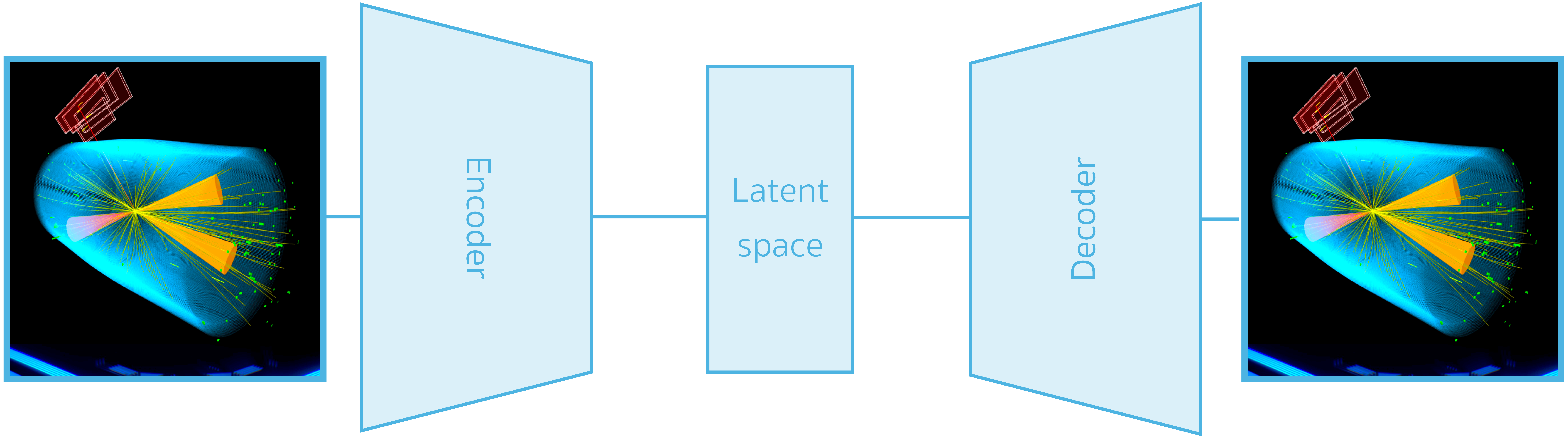
- Challenge: if new physics has an unexpected signature that doesn't align with existing triggers, precious BSM events may be discarded at trigger level
- Can we use unsupervised algorithms to detect non-SM-like anomalies?

APPLICATION: ANOMALY DETECTION AT 40 MHZ

Nat. Mach. Intell. 4, 154 (2022)

Data challenge: mpp-hep.github.io/ADC2021

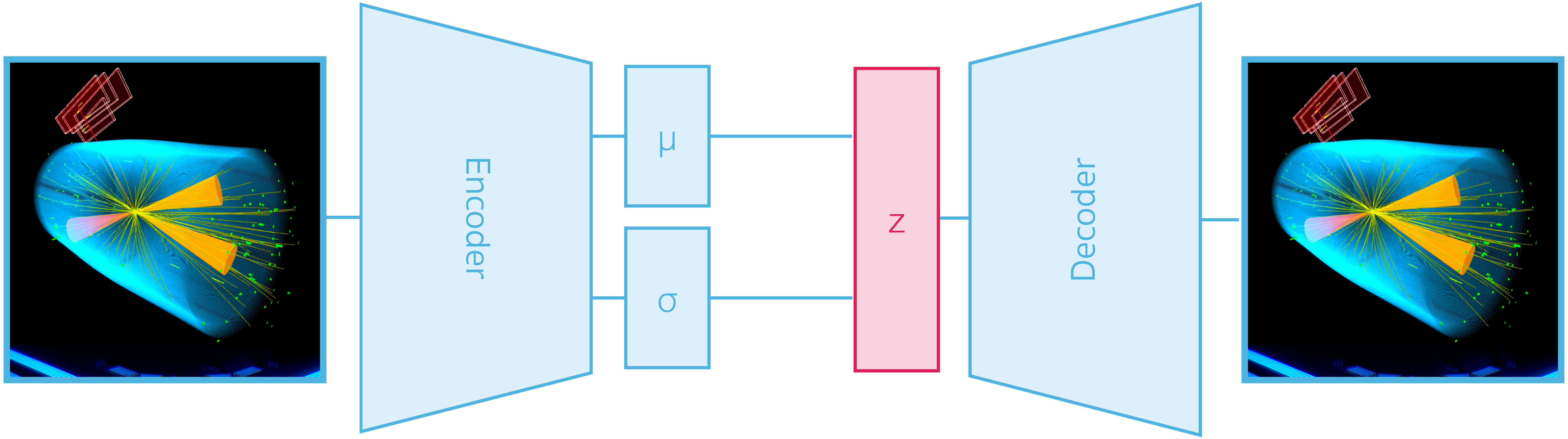
- Challenge: if new physics has an unexpected signature that doesn't align with existing triggers, precious BSM events may be discarded at trigger level
- Can we use unsupervised algorithms to detect non-SM-like anomalies?
  - Autoencoders (AEs): compress input to a smaller dimensional latent space then decompress and calculate difference



APPLICATION: ANOMALY DETECTION AT 40 MHZ

[Nat. Mach. Intell. 4, 154 (2022)](#)
Data challenge: mpp-hep.github.io/ADC2021

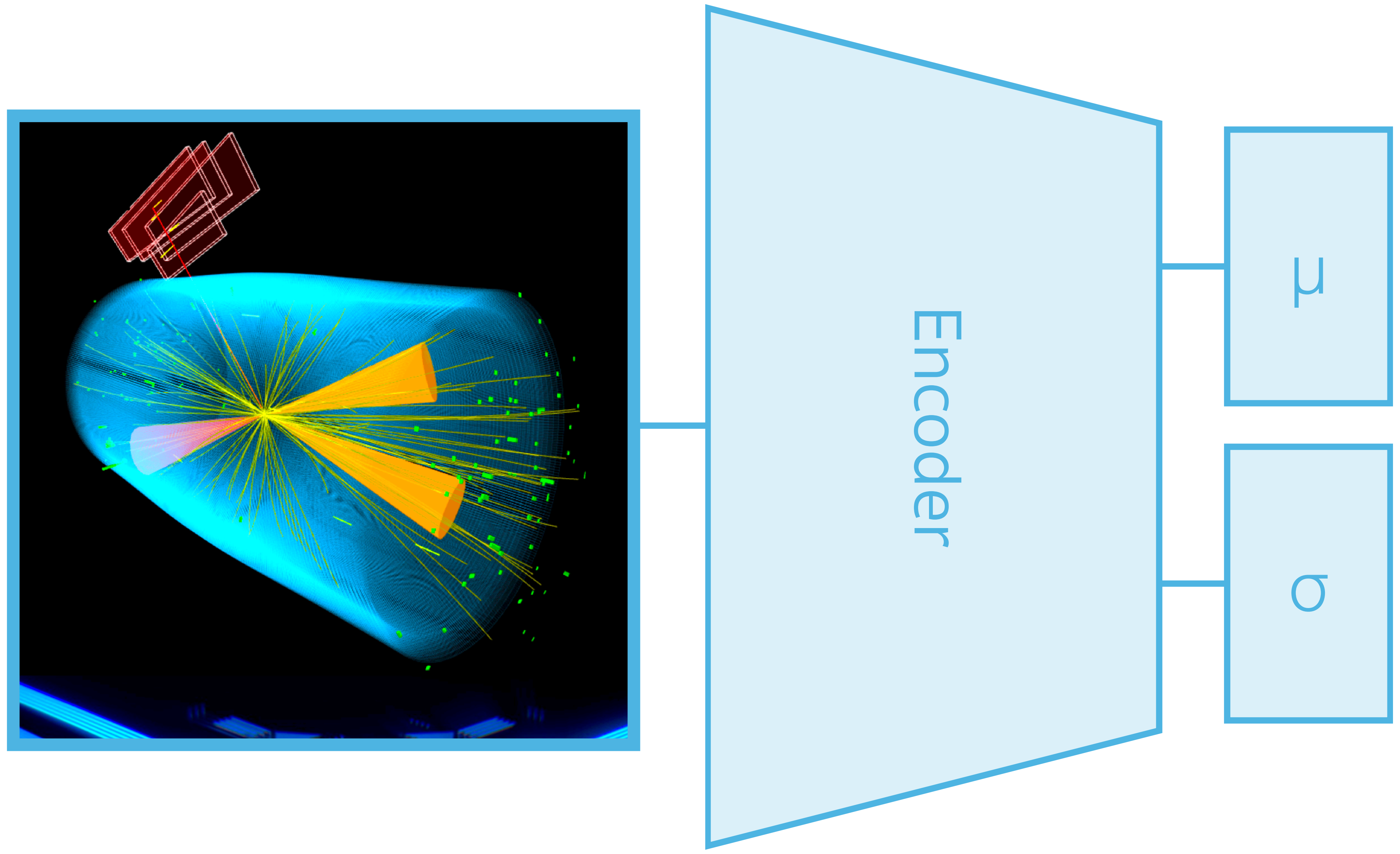
- Challenge: if new physics has an unexpected signature that doesn't align with existing triggers, precious BSM events may be discarded at trigger level
- Can we use unsupervised algorithms to detect non-SM-like anomalies?
  - Autoencoders (AEs): compress input to a smaller dimensional latent space then decompress and calculate difference
  - Variational autoencoders (VAEs): model the latent space as a probability distribution; possible to detect anomalies purely with latent space variables



APPLICATION: ANOMALY DETECTION AT 40 MHZ

Nat. Mach. Intell. 4, 154 (2022)
Data challenge: mpp-hep.github.io/ADC2021

- Challenge: if new physics has an unexpected signature that doesn't align with existing triggers, precious BSM events may be discarded at trigger level
- Can we use unsupervised algorithms to detect non-SM-like anomalies?
  - Autoencoders (AEs): compress input to a smaller dimensional latent space then decompress and calculate difference
  - Variational autoencoders (VAEs): model the latent space as a probability distribution; possible to detect anomalies purely with latent space variables



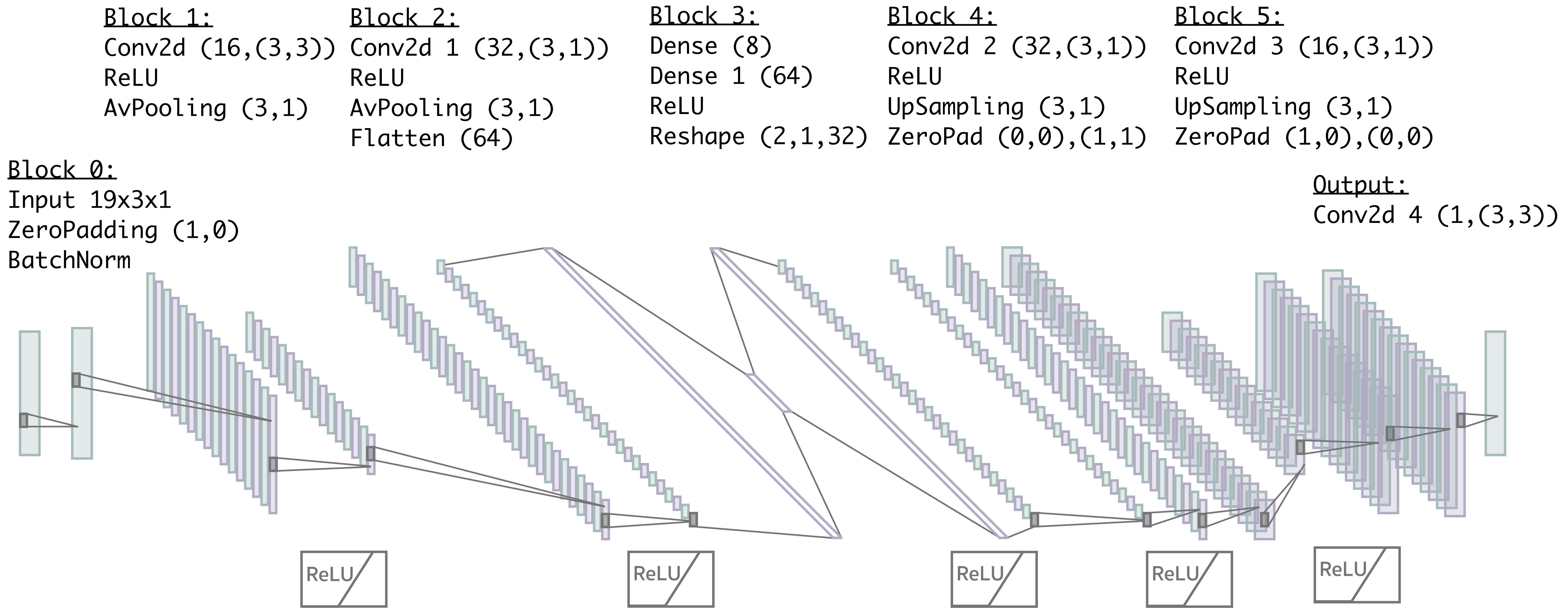
Key observation: Can build an anomaly score from the latent space of VAE directly! No need to run decoder!

$$R_z = \sum_i \frac{\mu_i^2}{\sigma_i^2}$$

FPGA IMPLEMENTATION

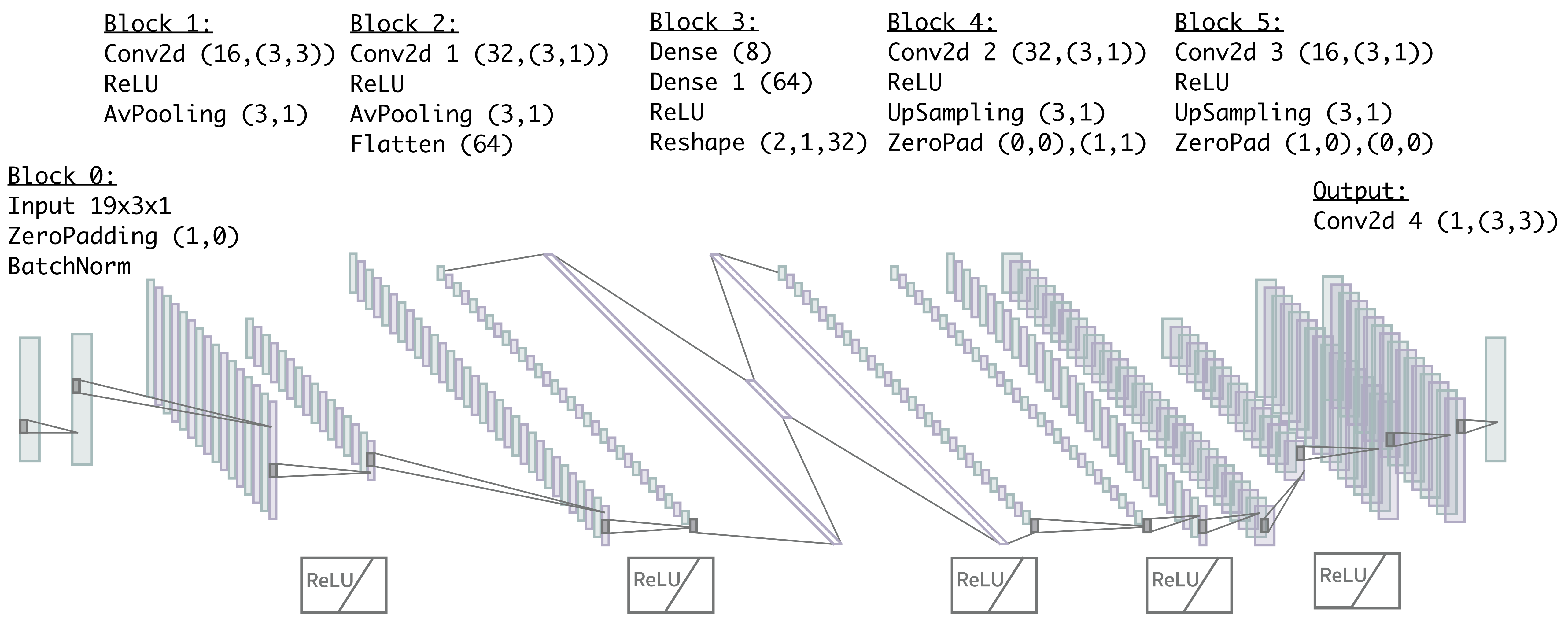
Nat. Mach. Intell. 4, 154 (2022)

Data challenge: mpp-hep.github.io/ADC2021



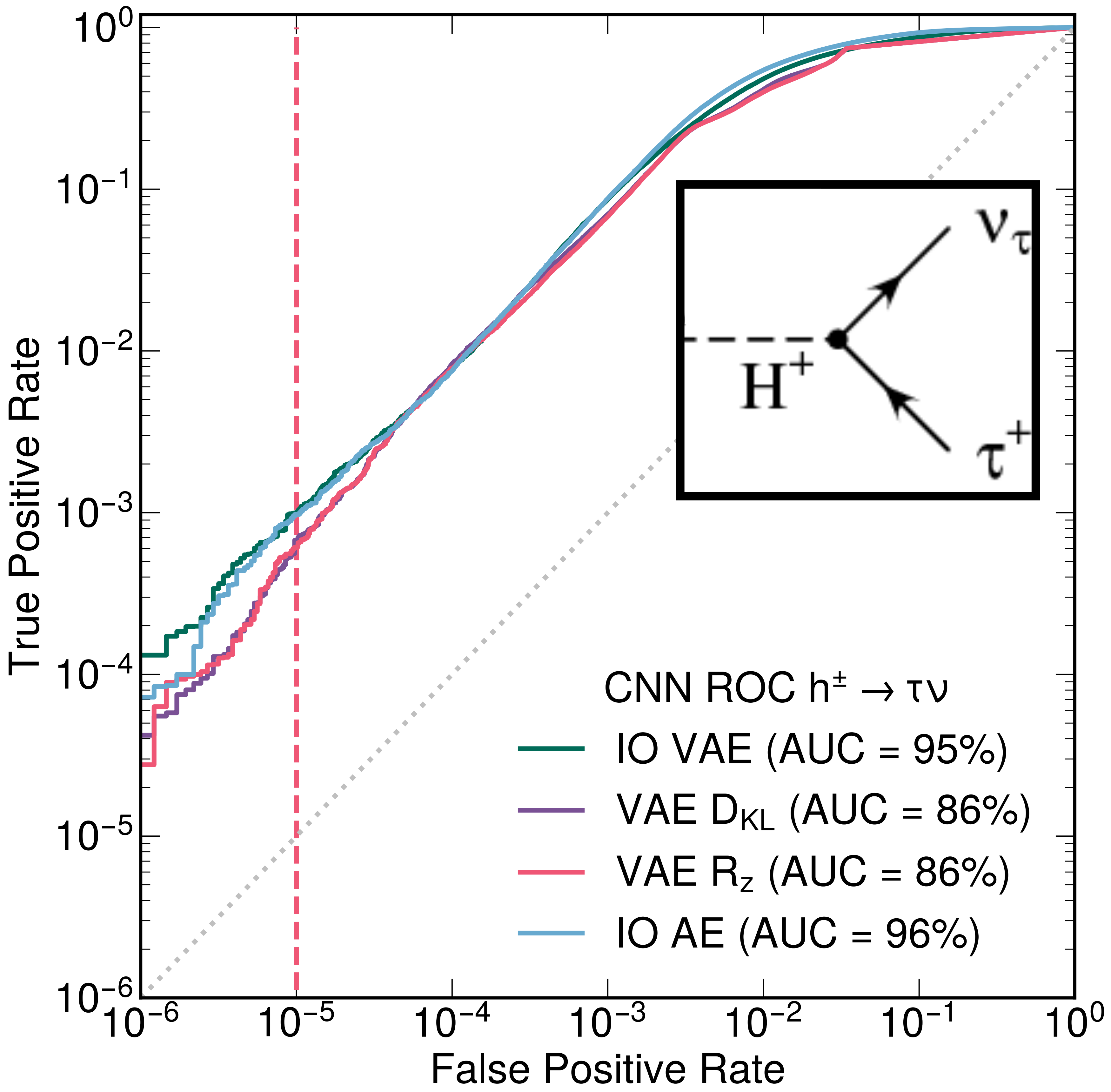
FPGA IMPLEMENTATION

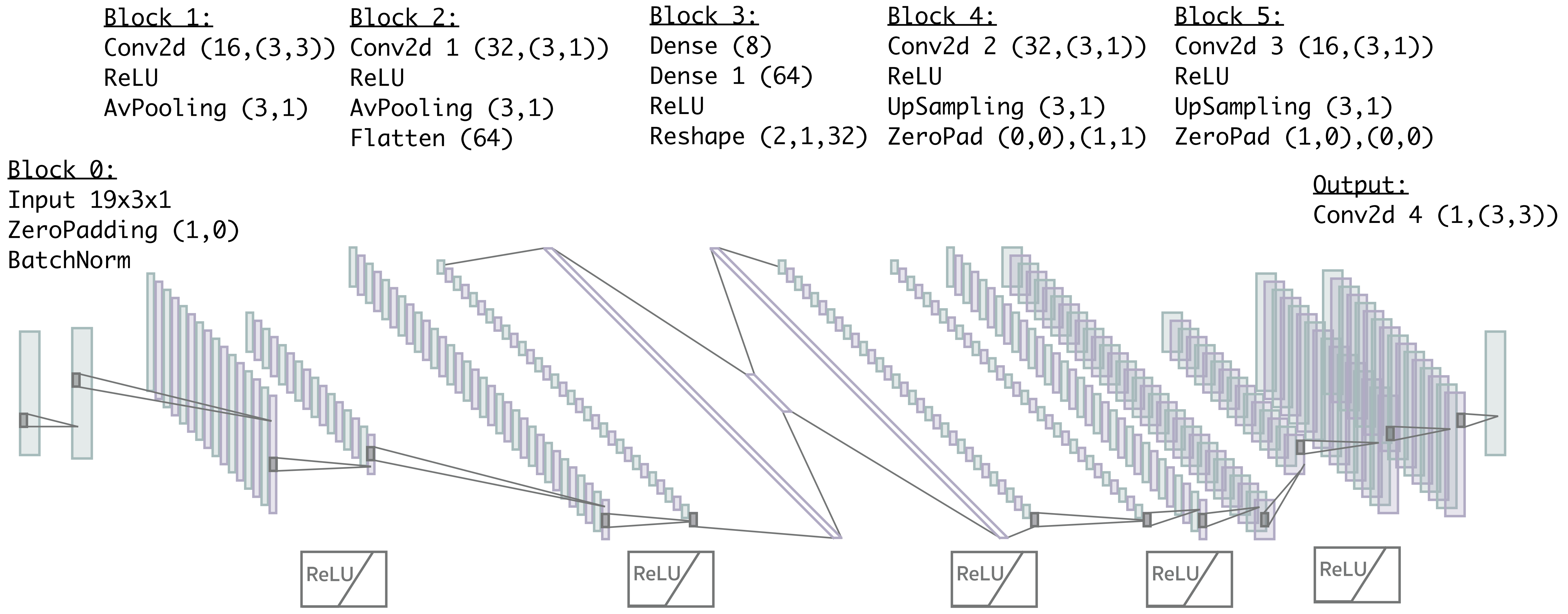
- CNNs as the basis for (V)AEs for anomaly detection



FPGA IMPLEMENTATION

- CNNs as the basis for (V)AEs for anomaly detection
- Good anomaly detection performance for unseen signals ($LQ \to b\tau$, $A \to 4l$, $\mathbf{h^\pm \to \tau\nu}$, $h^0 \to \tau\tau$)

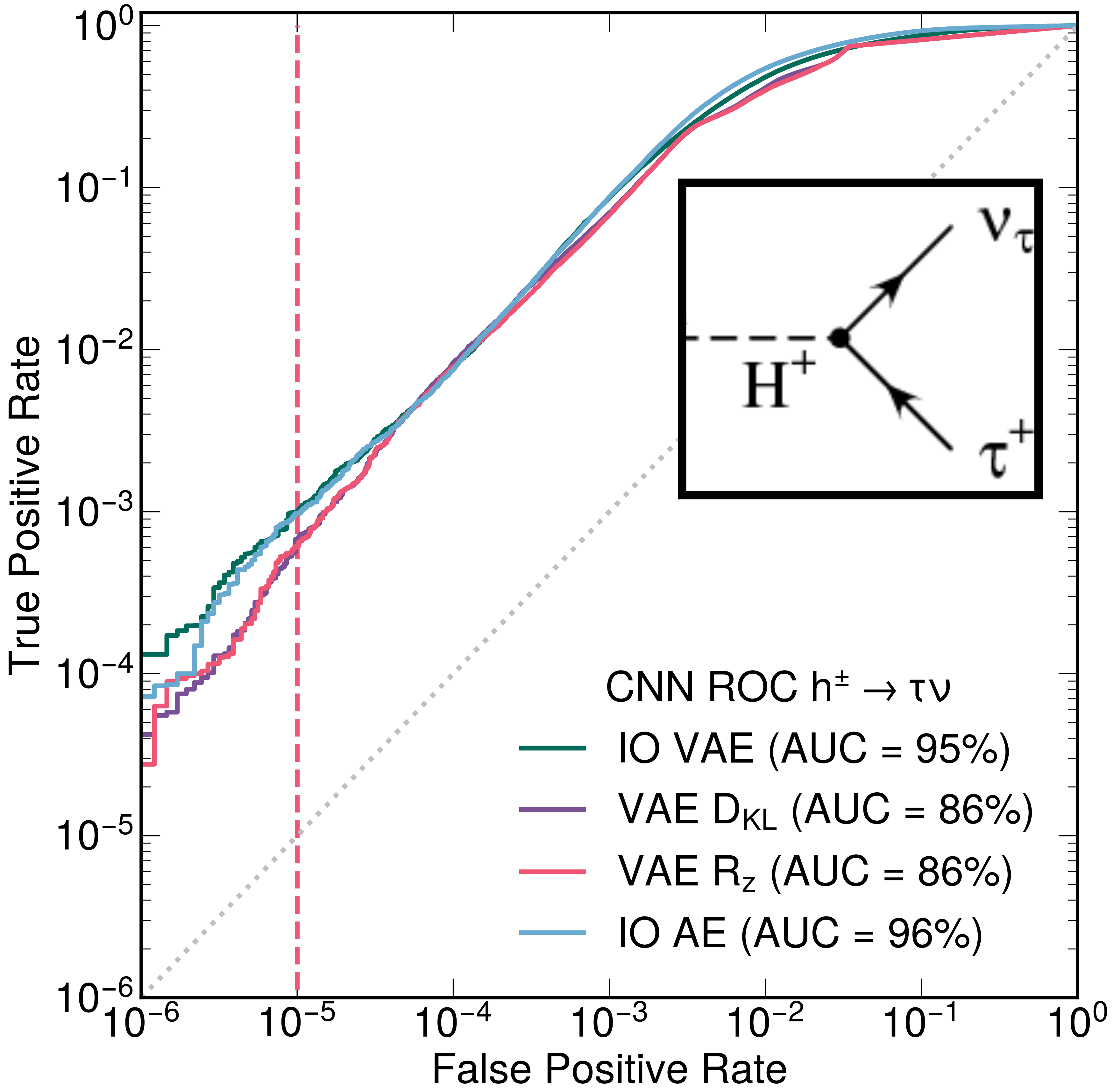


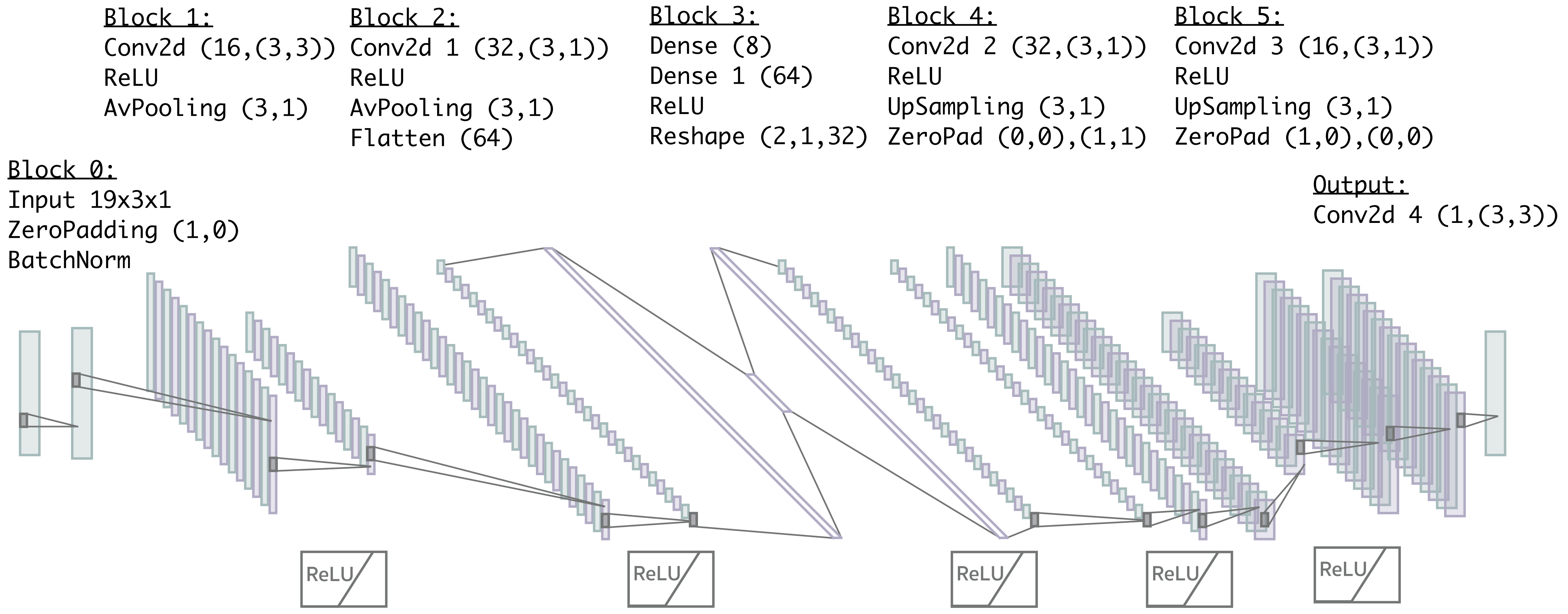


FPGA IMPLEMENTATION

Nat. Mach. Intell. 4, 154 (2022)
Data challenge: mpp-hep.github.io/ADC2021

- CNNs as the basis for (V)AEs for anomaly detection
- Good anomaly detection performance for unseen signals ($LQ \to b\tau$, $A \to 4l$, $\mathbf{h^{\pm} \to \tau\nu}$, $h^0 \to \tau\tau$)
- **VAE** fits in latency and resource requirements for HL-LHC!





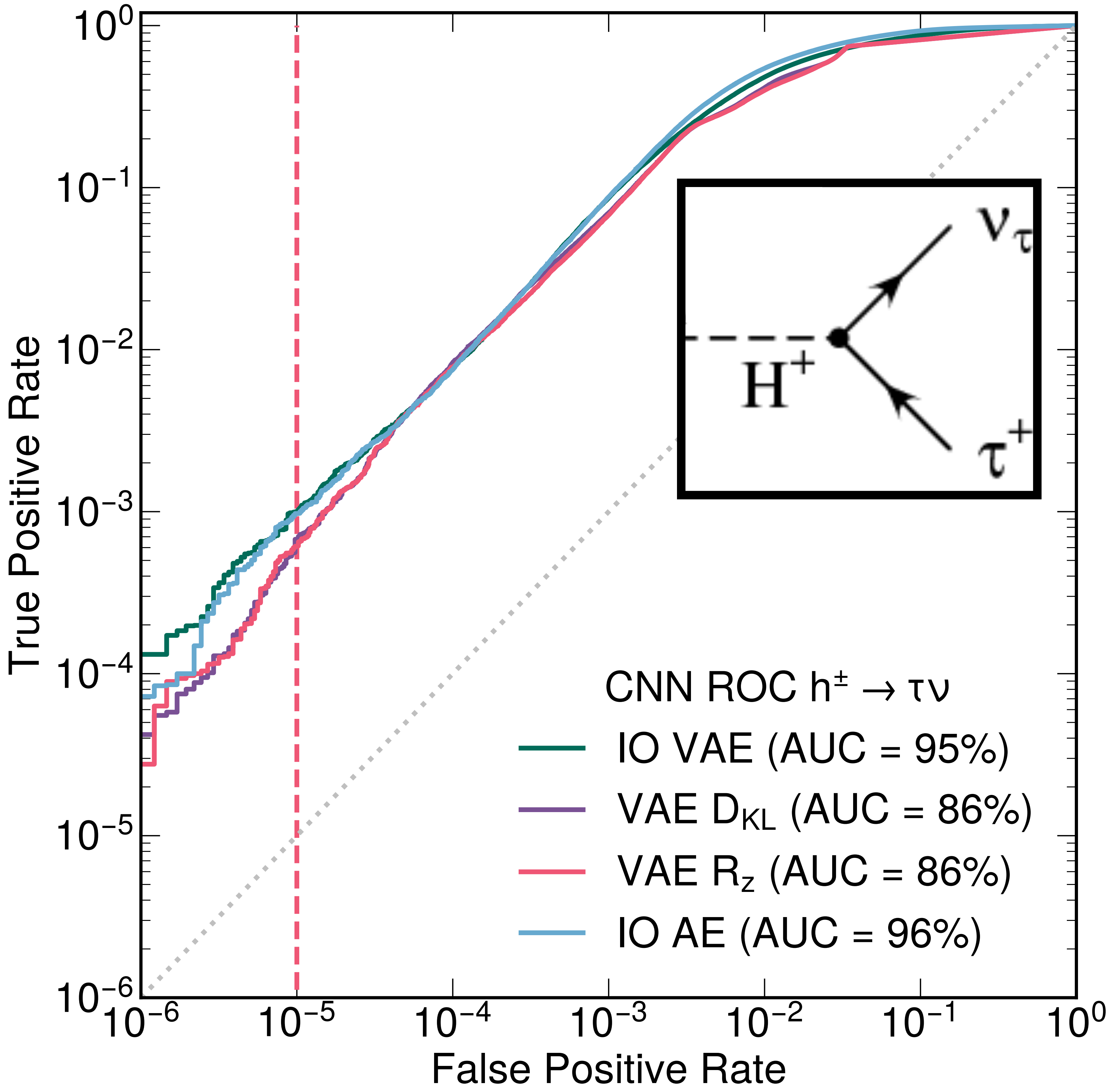
Model	DSP [%]	LUT [%]	FF [%]	BRAM [%]	Latency [ns]	II [ns]	AUC [%]	TPR @ FPR=10^{-5}
CNN VAE R_z	10	**12**	**4**	**2**	**365**	**115**	86	0.06%

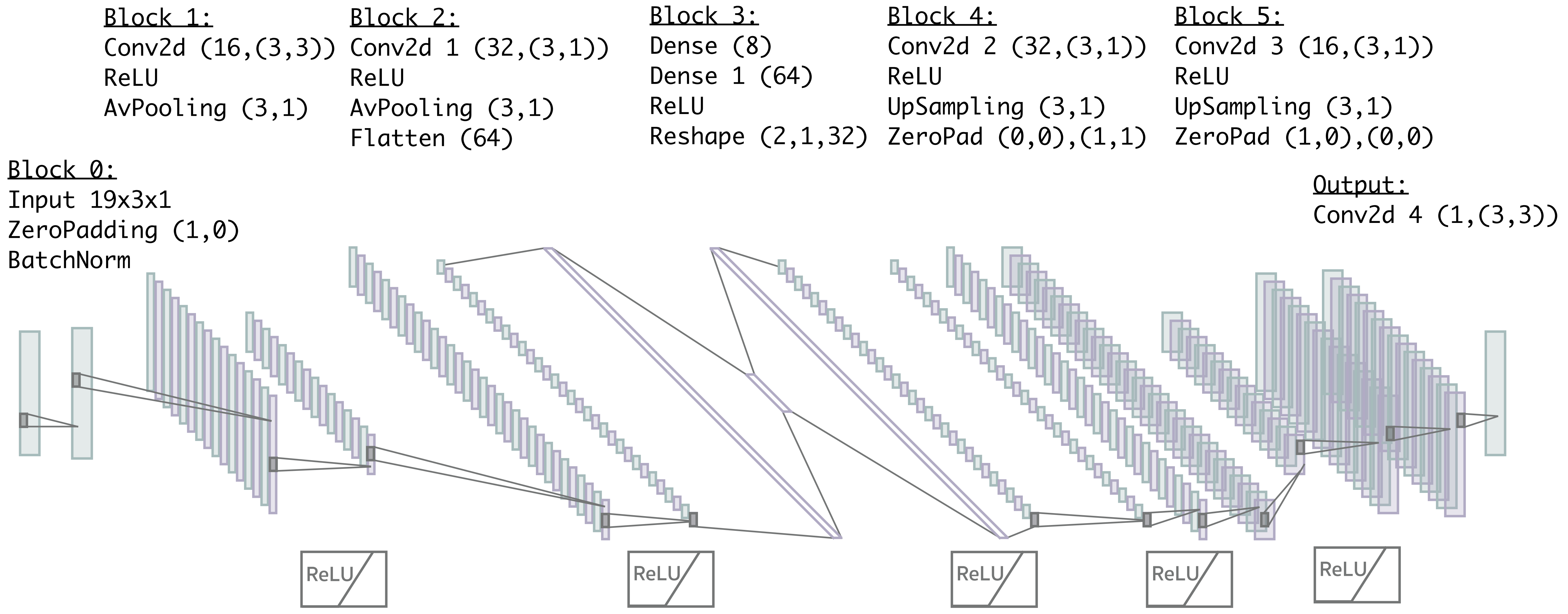
FPGA IMPLEMENTATION

Nat. Mach. Intell. 4, 154 (2022)
Data challenge: mpp-hep.github.io/ADC2021

- CNNs as the basis for (V)AEs for anomaly detection
- Good anomaly detection performance for unseen signals (LQ → bτ, A → 4l, **h± → τν**, h⁰ → ττ)
- **VAE** fits in latency and resource requirements for HL-LHC!

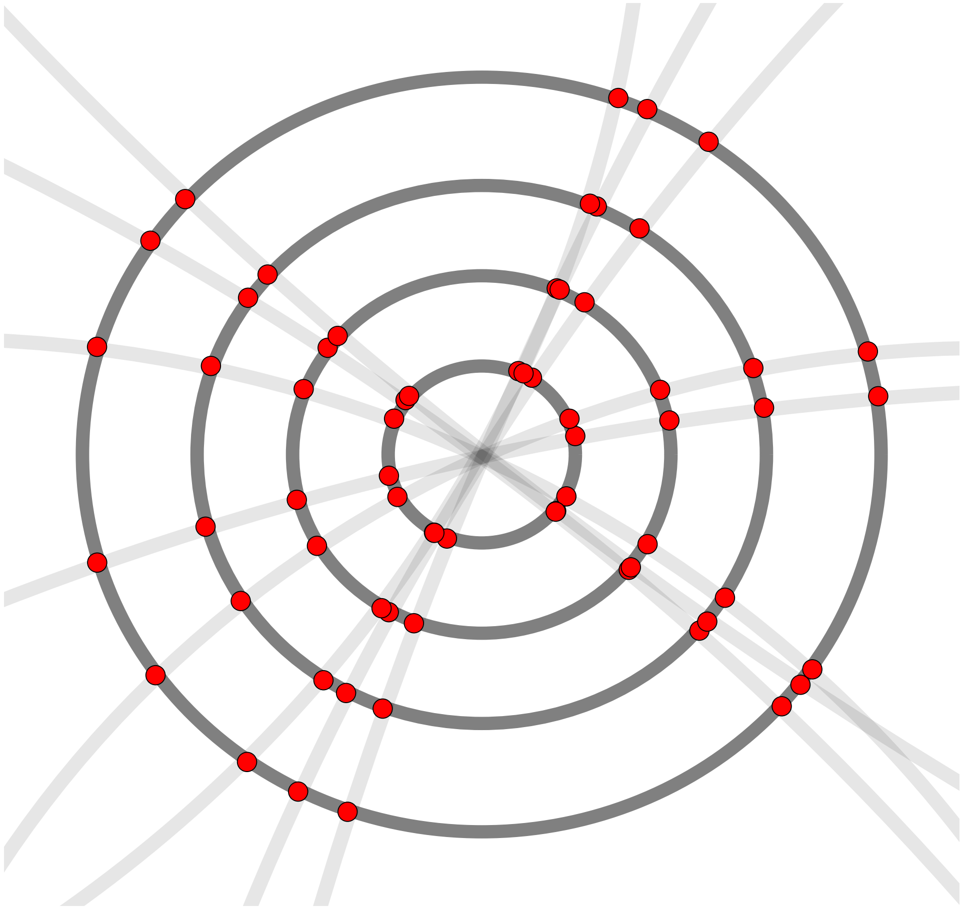
Stay tuned for Run 3…





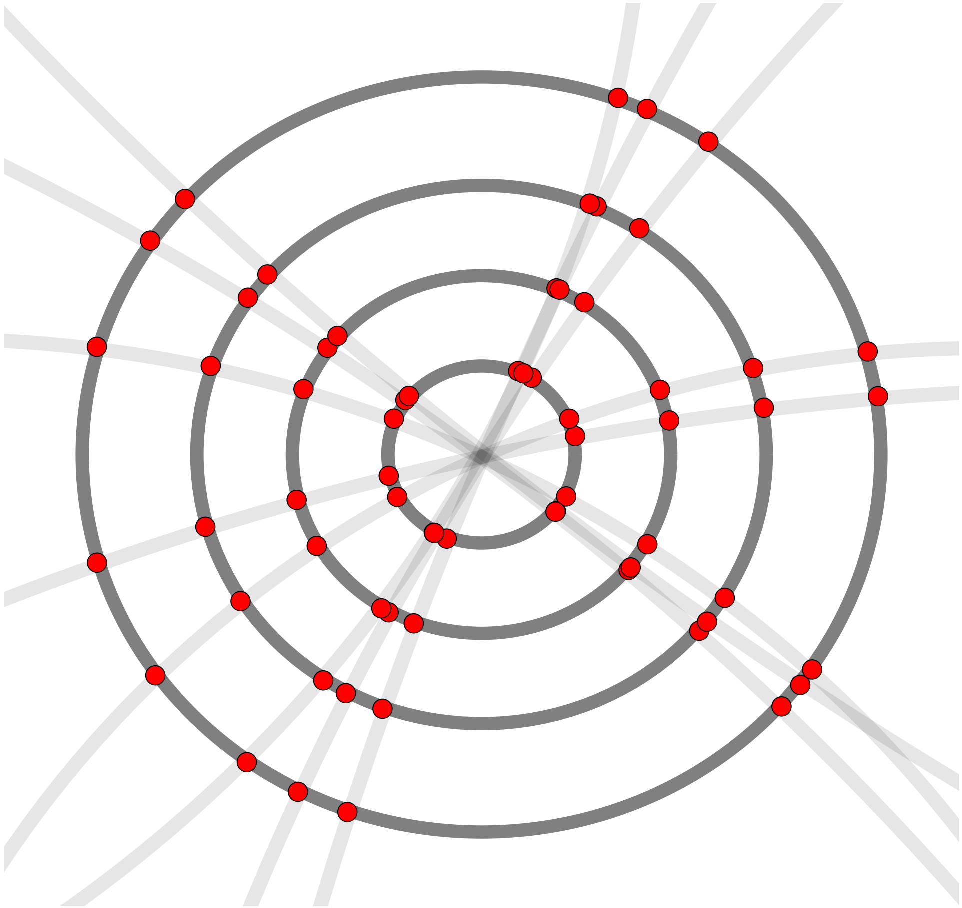
Model	DSP [%]	LUT [%]	FF [%]	BRAM [%]	Latency [ns]	II [ns]	AUC [%]	TPR @ FPR=10^{-5}
CNN VAE R_z	10	**12**	**4**	**2**	**365**	**115**	86	0.06%

APPLICATION: GRAPH NEURAL NETWORK TRACKING IN FPGAS



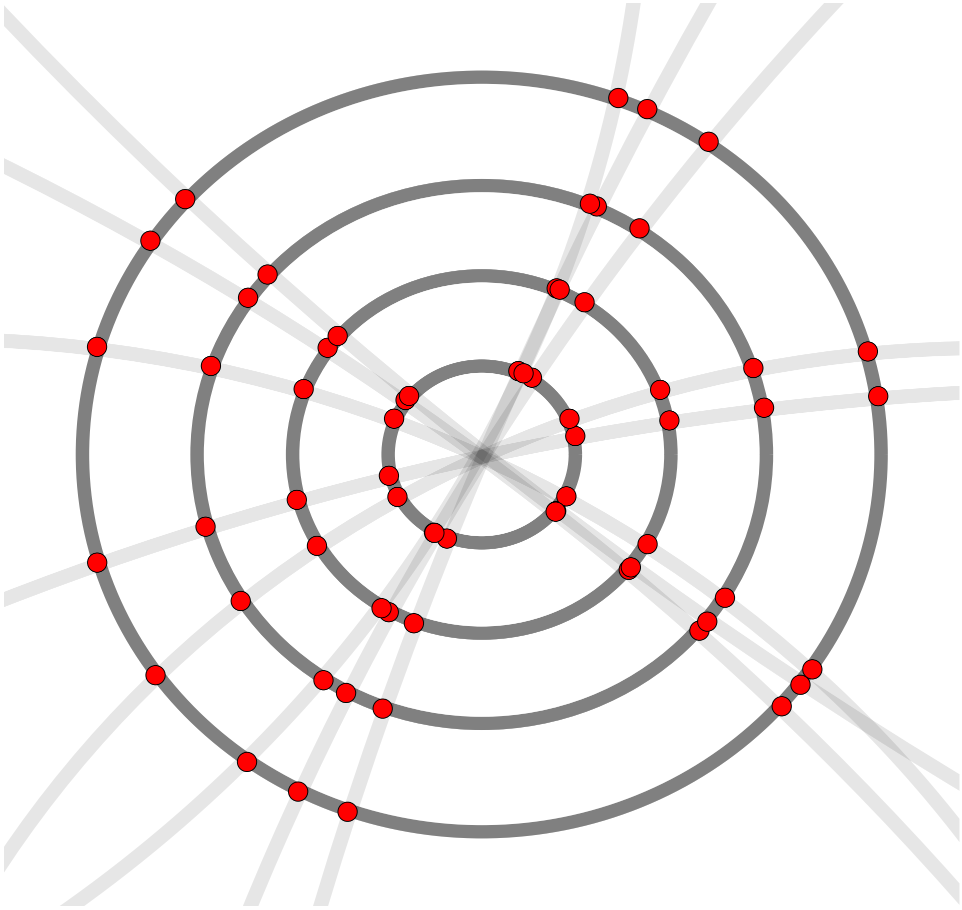
APPLICATION: GRAPH NEURAL NETWORK TRACKING IN FPGAS

- Traditional tracking algorithms scale quadratically with the number of hits



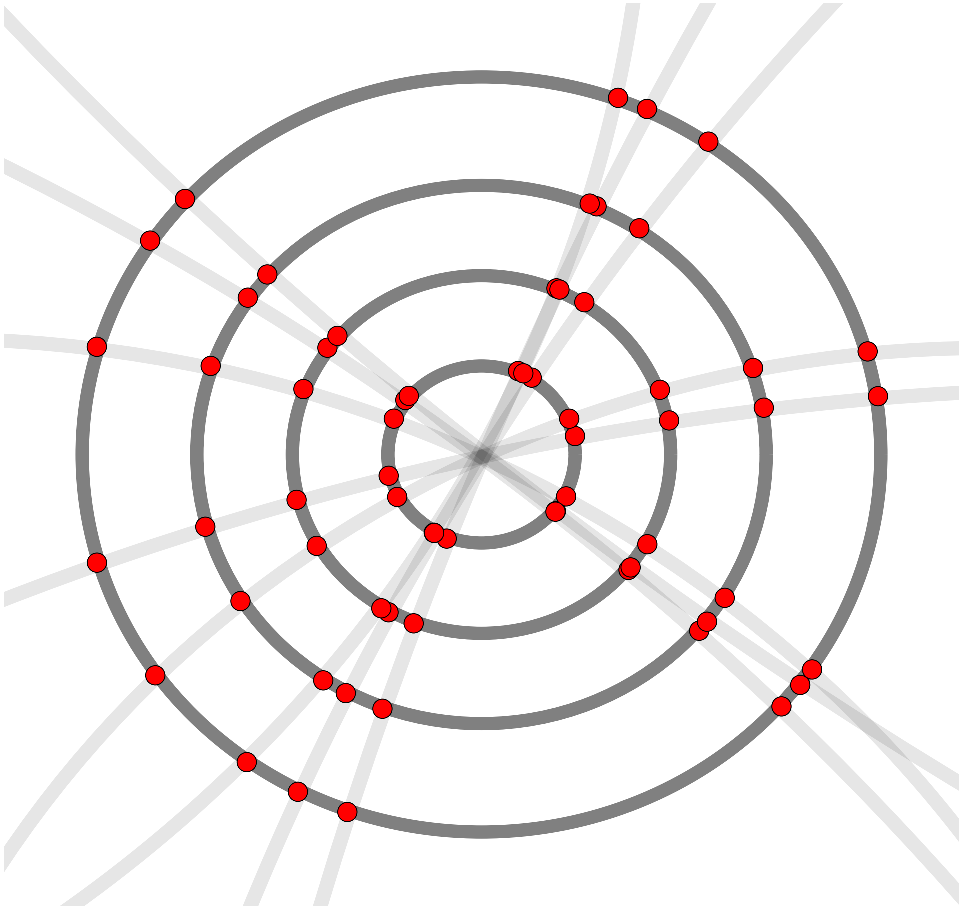
APPLICATION: GRAPH NEURAL NETWORK TRACKING IN FPGAS

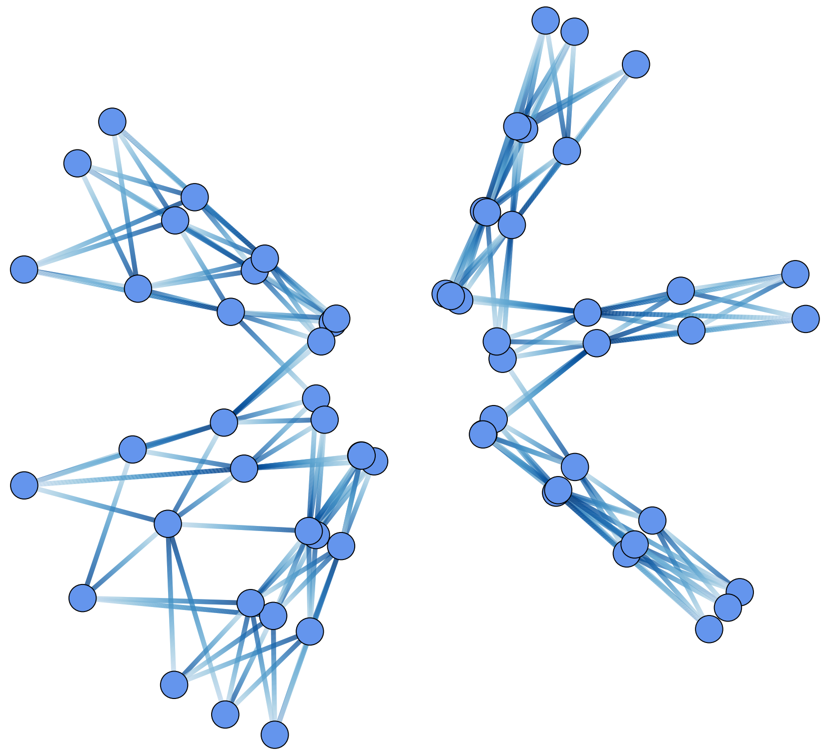
- Traditional tracking algorithms scale quadratically with the number of hits
- New algorithms (based on **graph neural networks**) may be able to do better

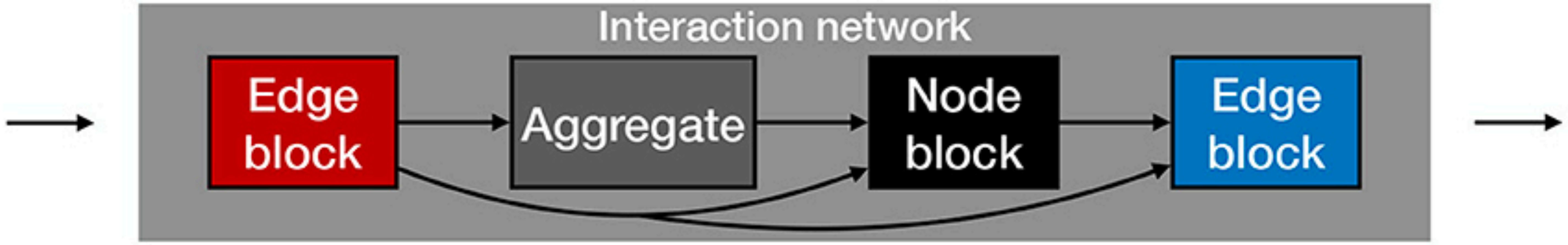


APPLICATION: GRAPH NEURAL NETWORK TRACKING IN FPGAS

- Traditional tracking algorithms scale quadratically with the number of hits
- New algorithms (based on **graph neural networks**) may be able to do better
- Proof of concept study: use GNN to classify good track segments (edges)

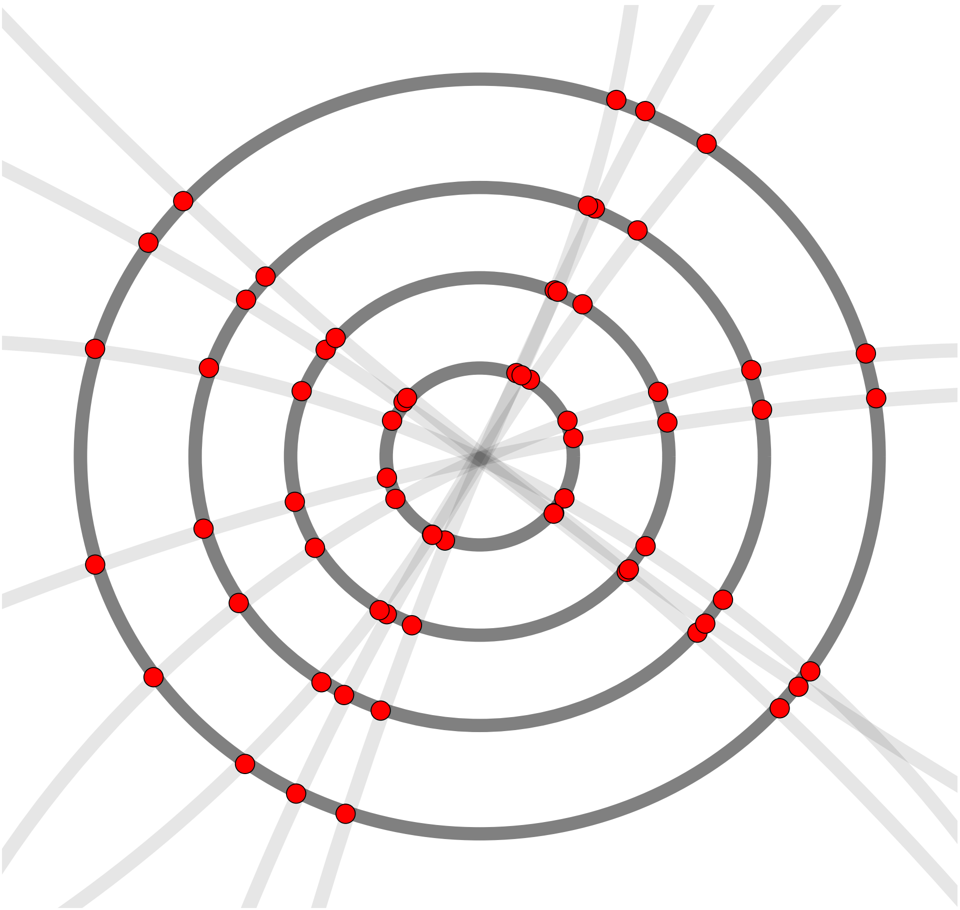


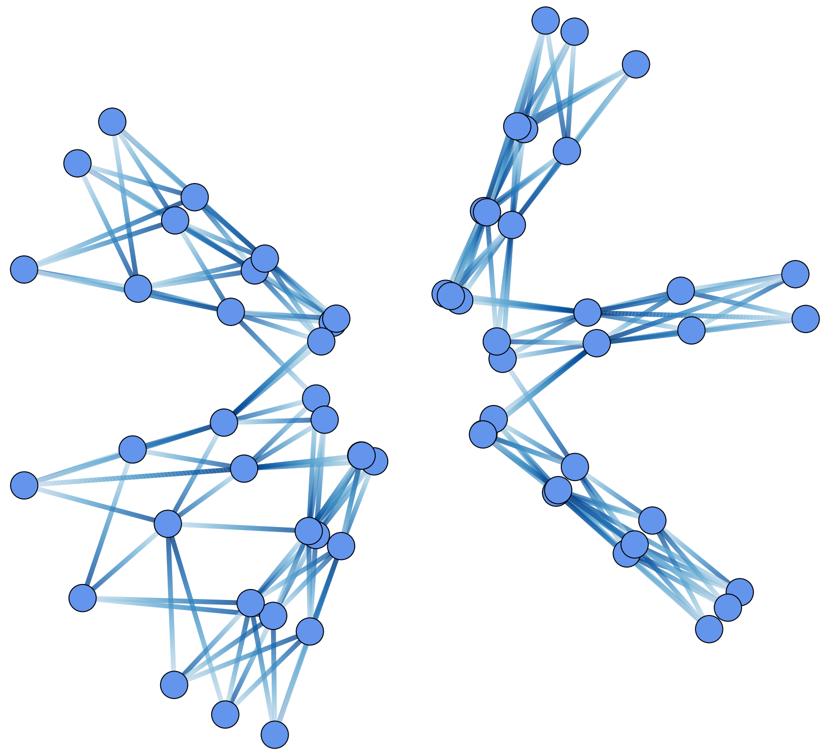


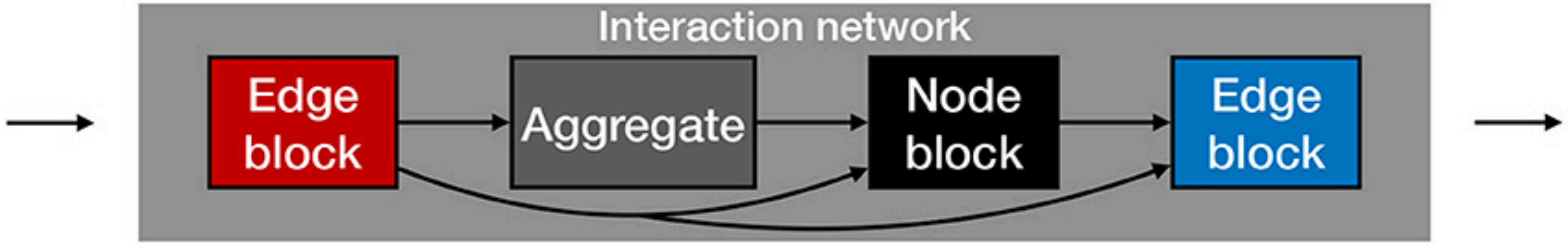


APPLICATION: GRAPH NEURAL NETWORK TRACKING IN FPGAS

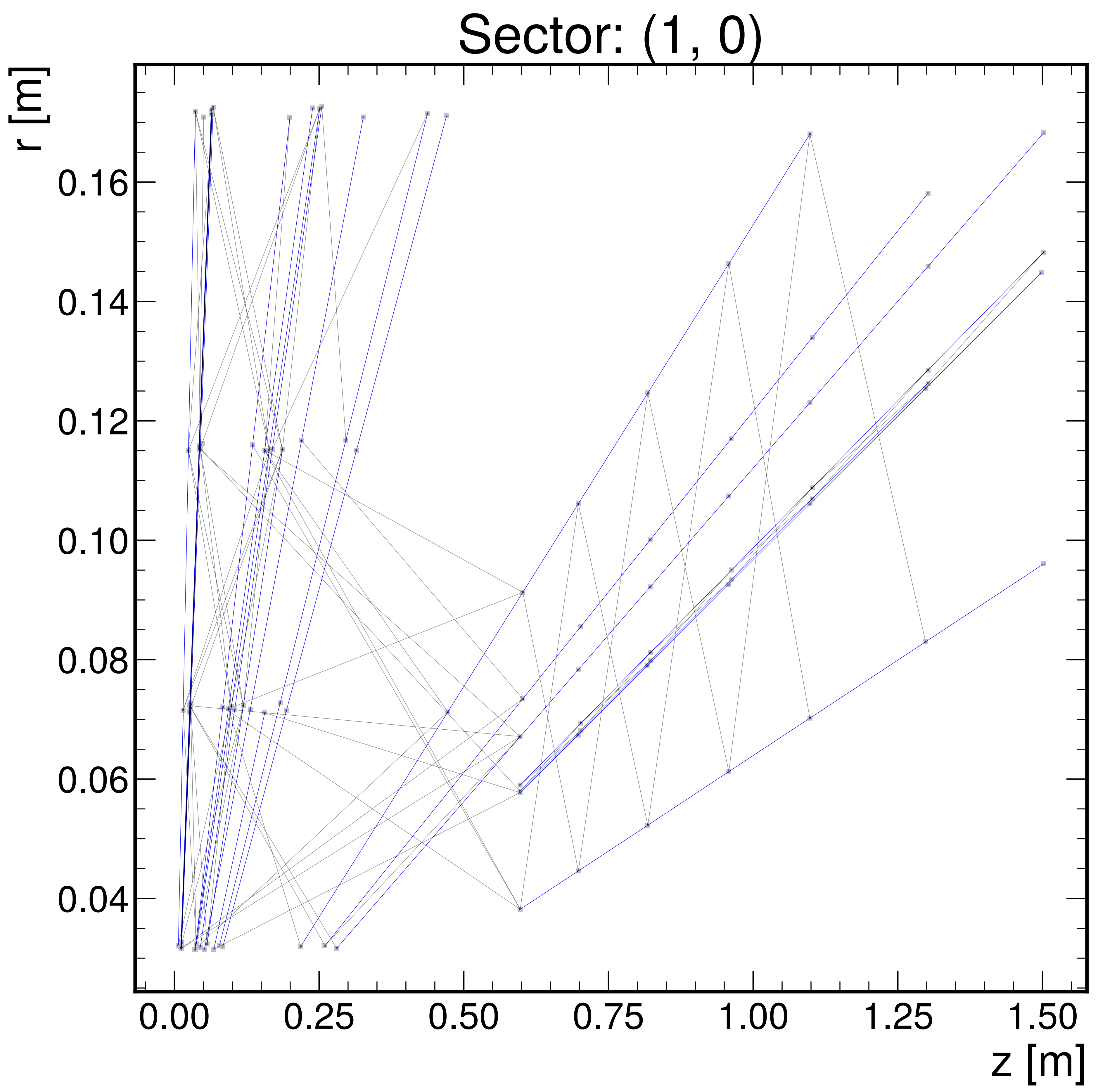
- Traditional tracking algorithms scale quadratically with the number of hits
- New algorithms (based on **graph neural networks**) may be able to do better
- Proof of concept study: use GNN to classify good track segments (edges)
    - Can this fit on an FPGA?



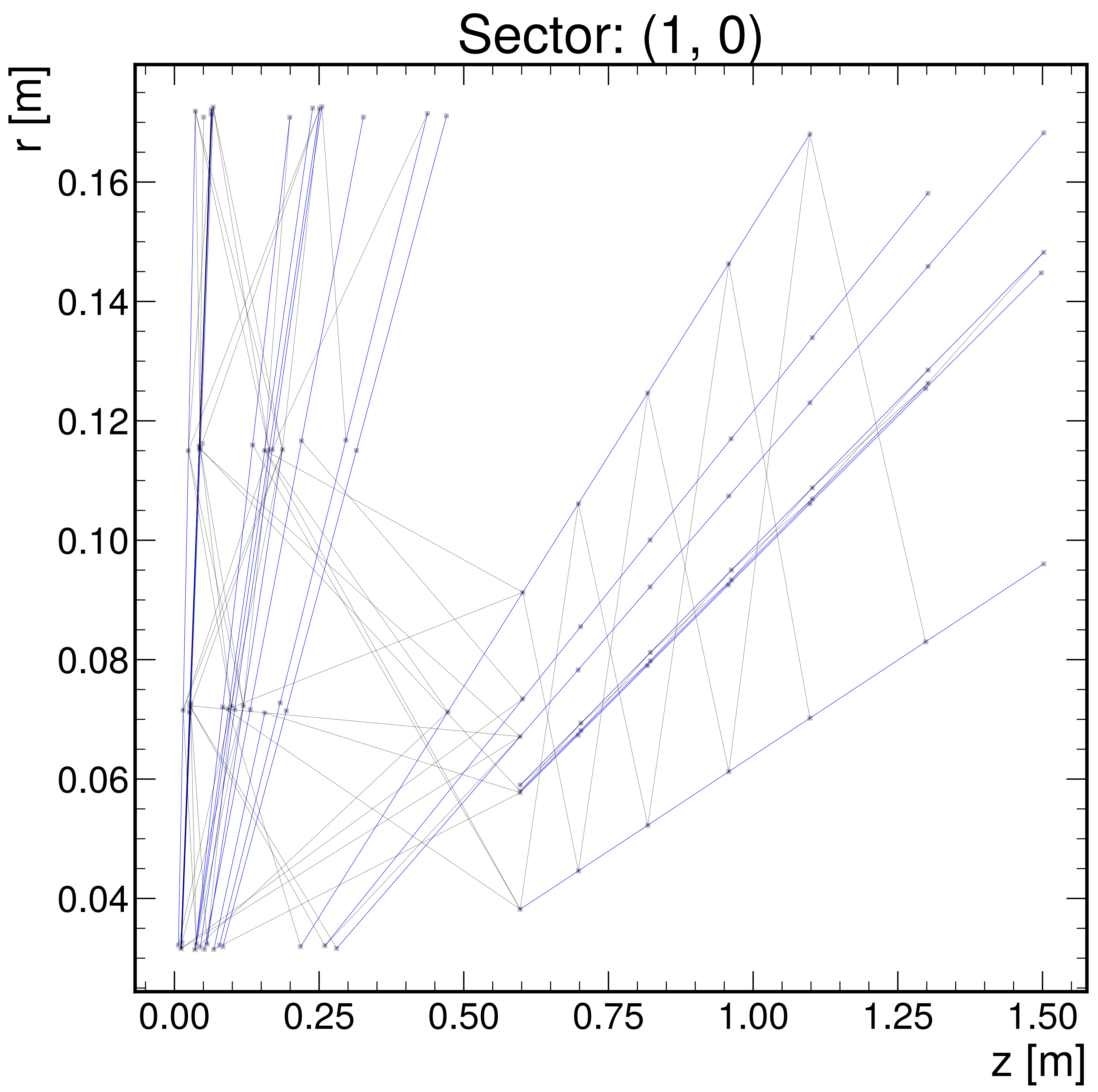




GNN EDGE CLASSIFICATION PERFORMANCE



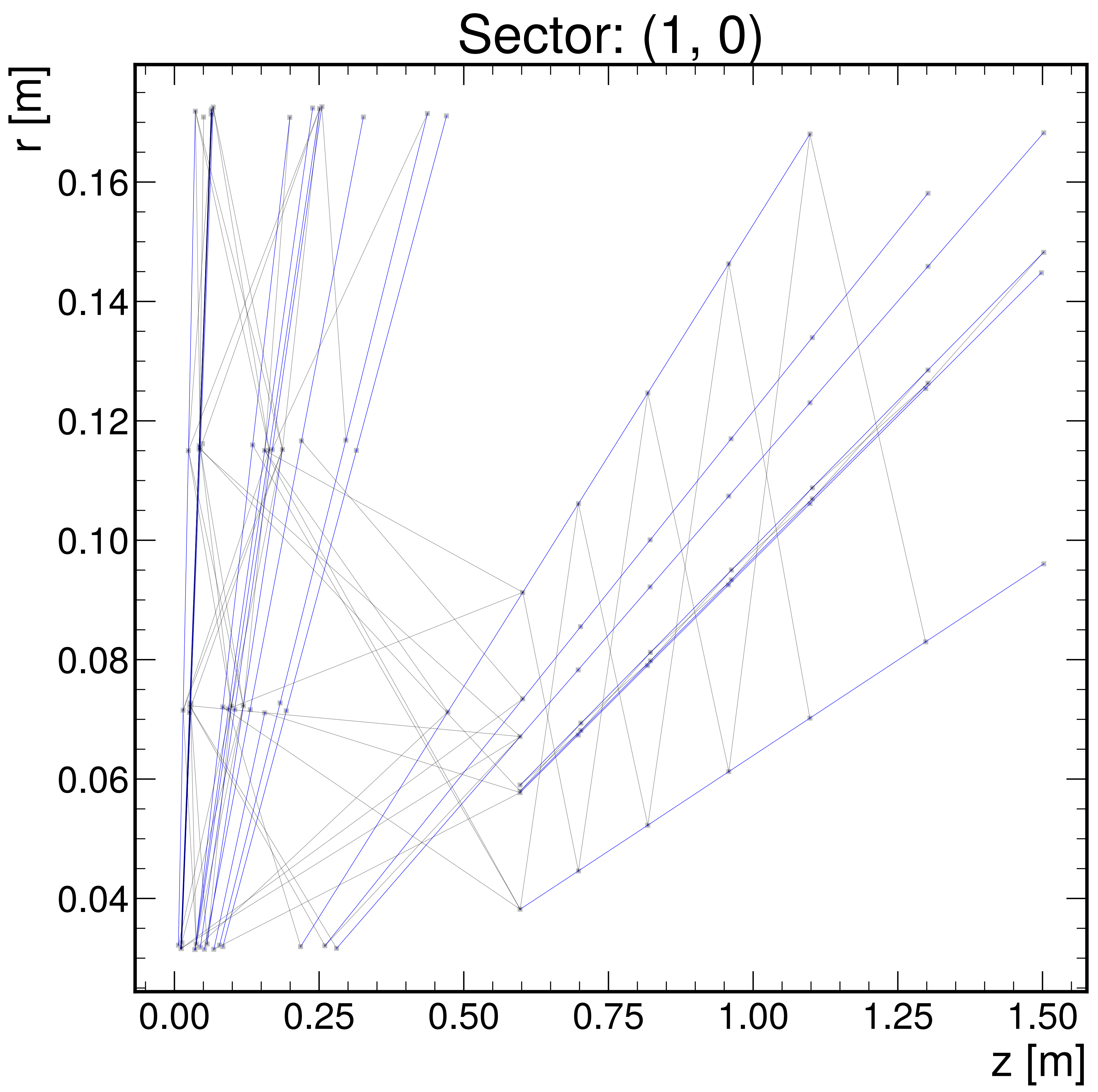
GNN EDGE CLASSIFICATION PERFORMANCE

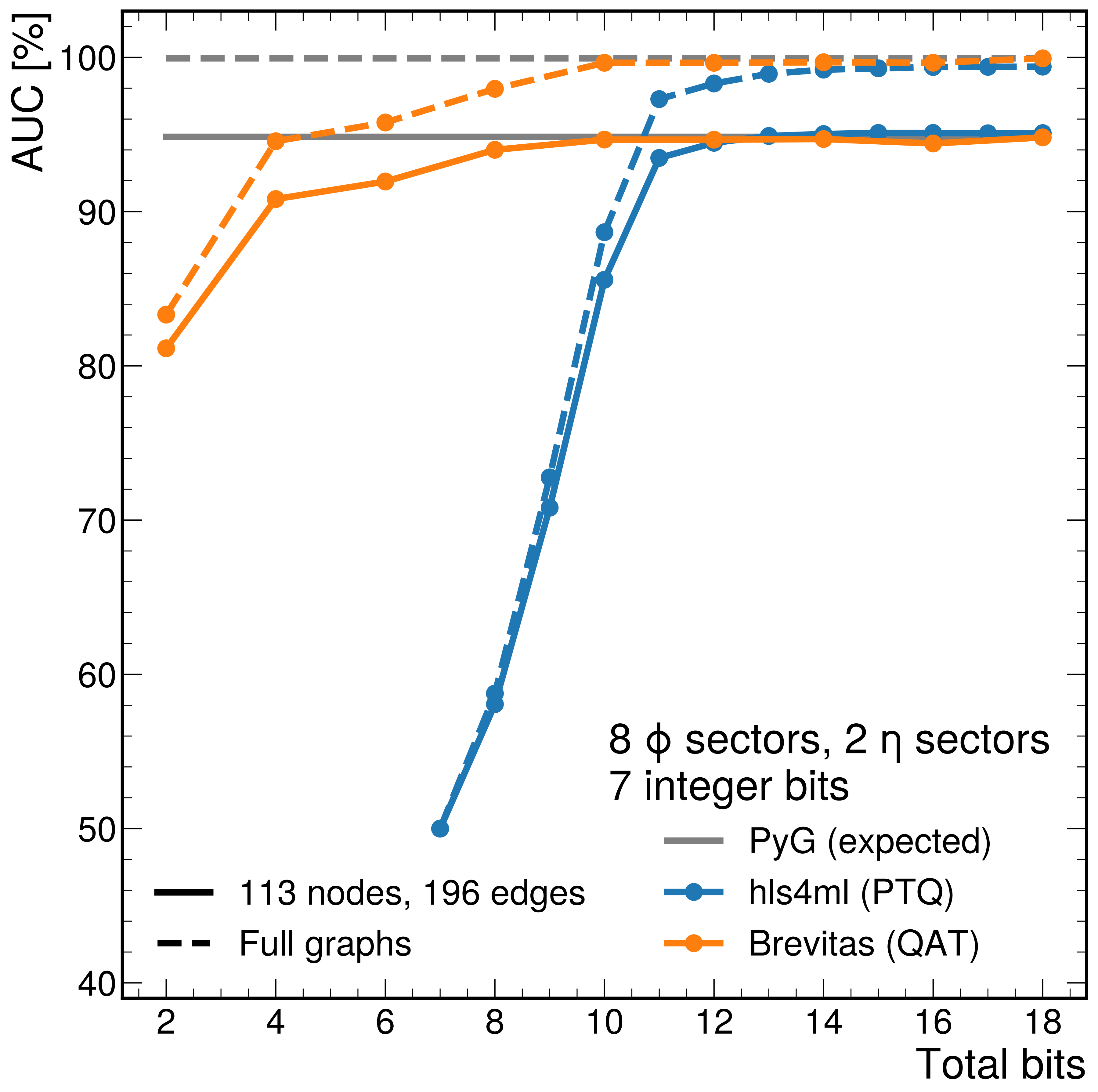


- Build realistic (segmented) graphs for L1 trigger applications

GNN EDGE CLASSIFICATION PERFORMANCE

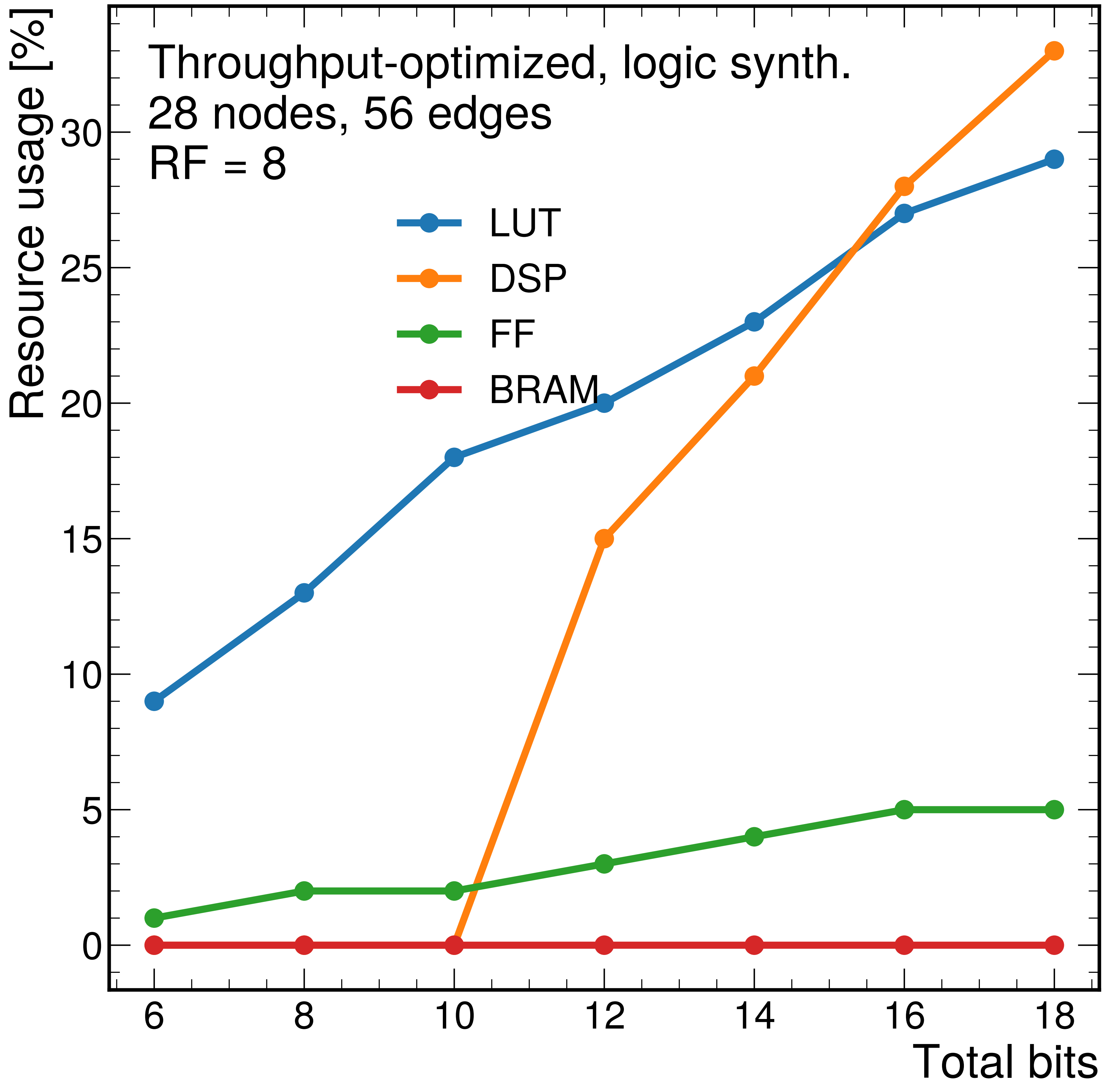
Front. Big Data 5, 828666 (2022)



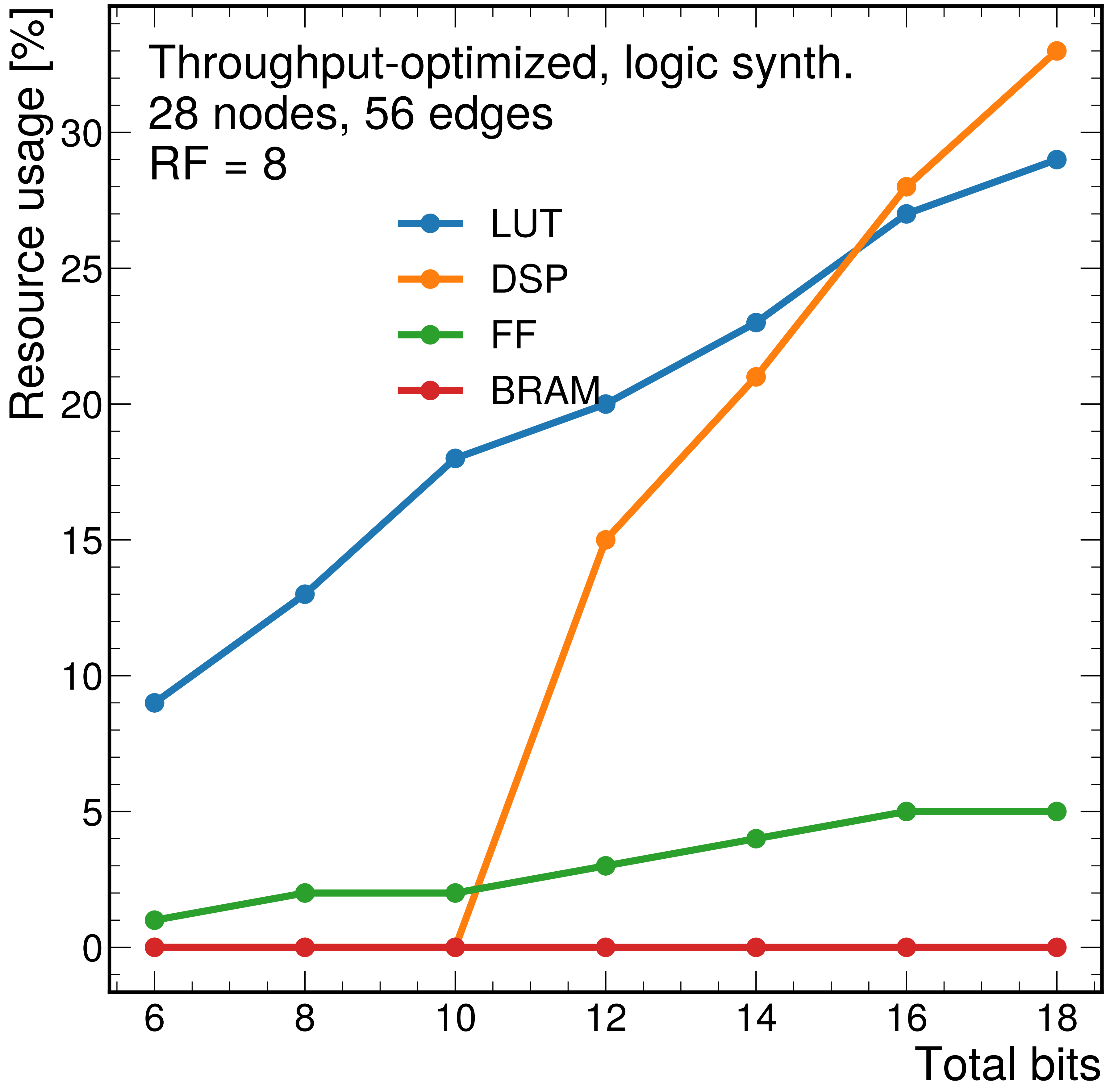


- Build realistic (segmented) graphs for L1 trigger applications
- ≤8-bit quantized GNN can achieve good edge classification performance

GNN FPGA IMPLEMENTATION

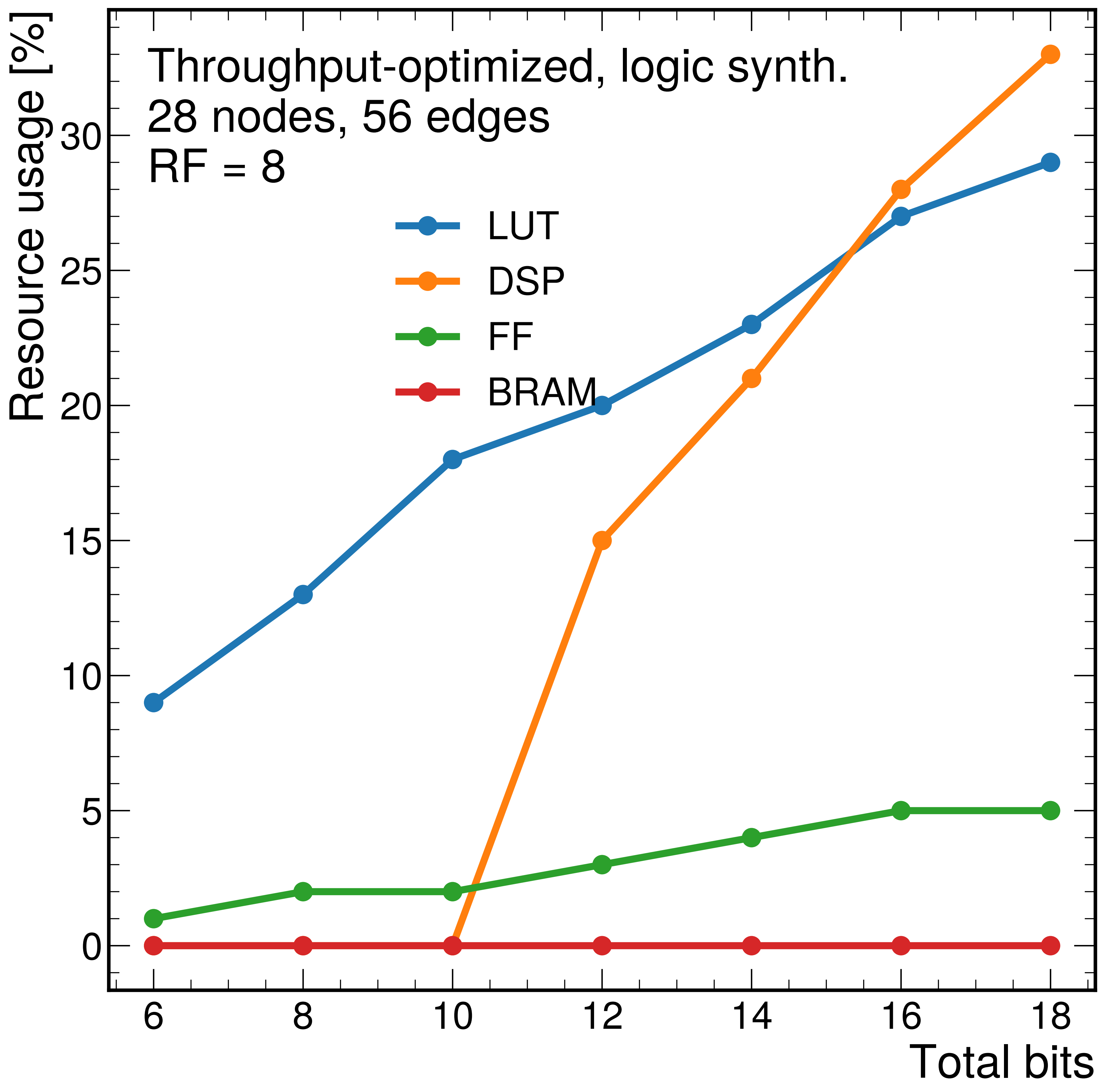


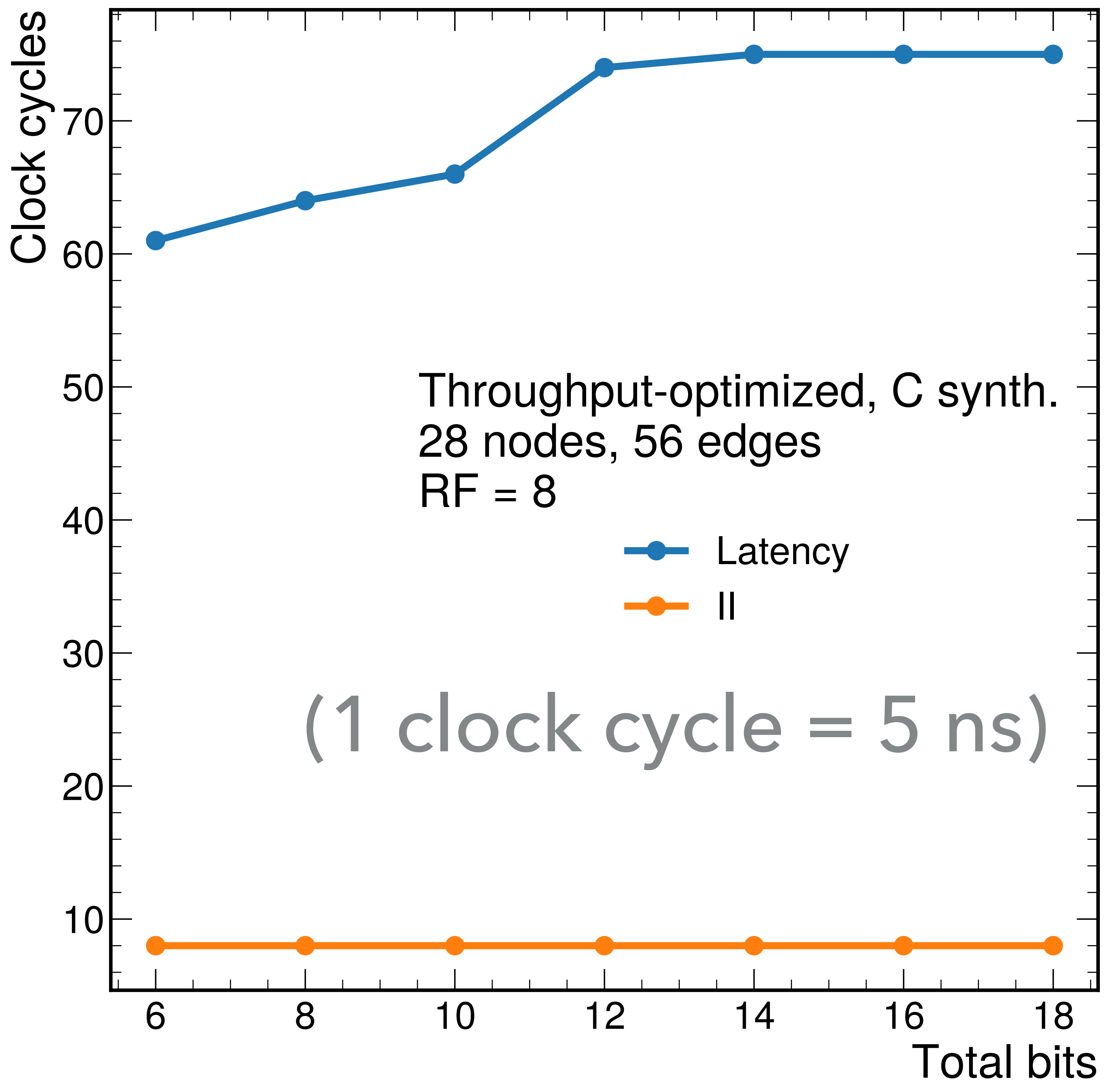
GNN FPGA IMPLEMENTATION



- Small graphs (~30 nodes, ~60 edges) easily fit on 1 FPGA

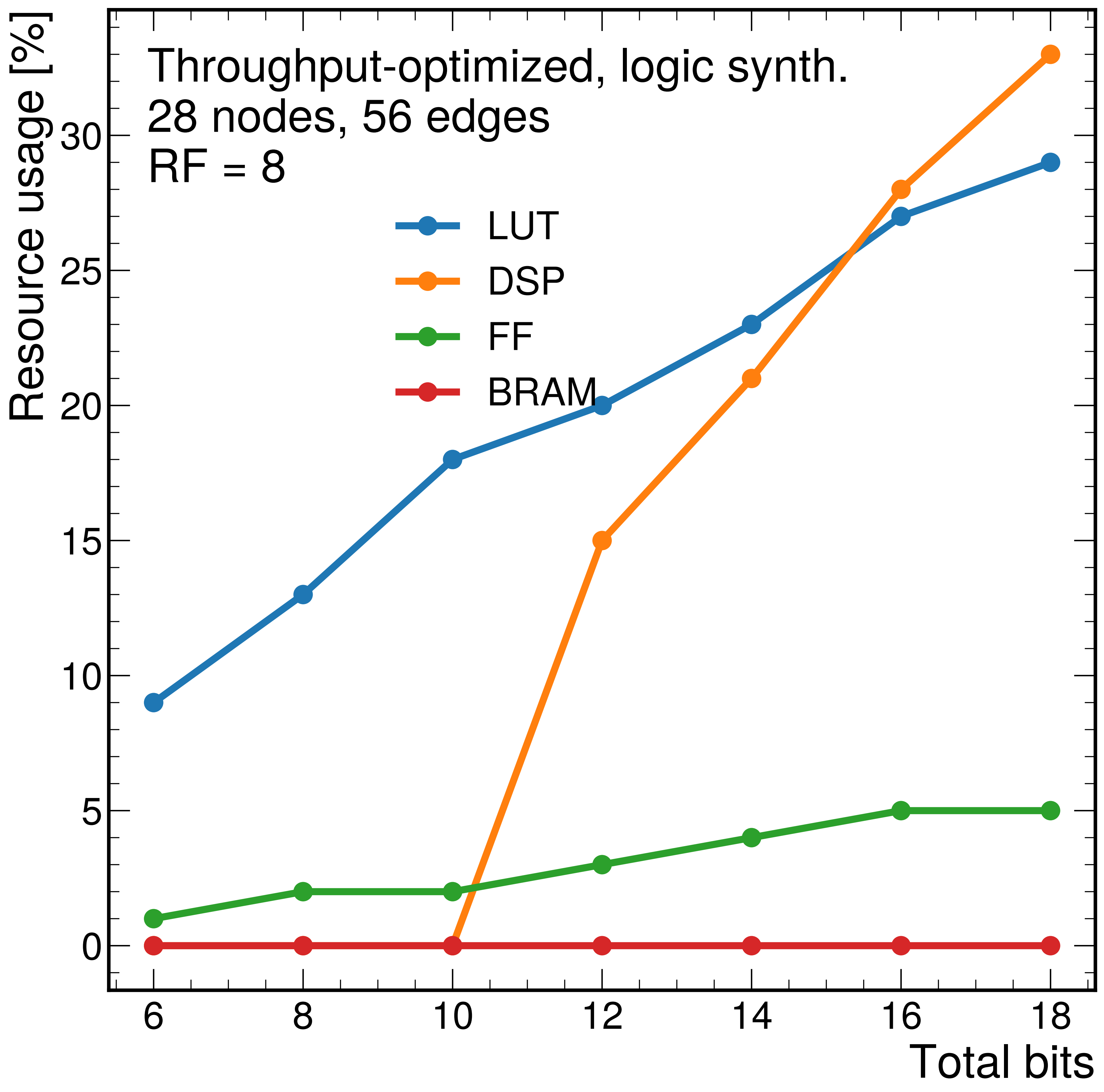
GNN FPGA IMPLEMENTATION

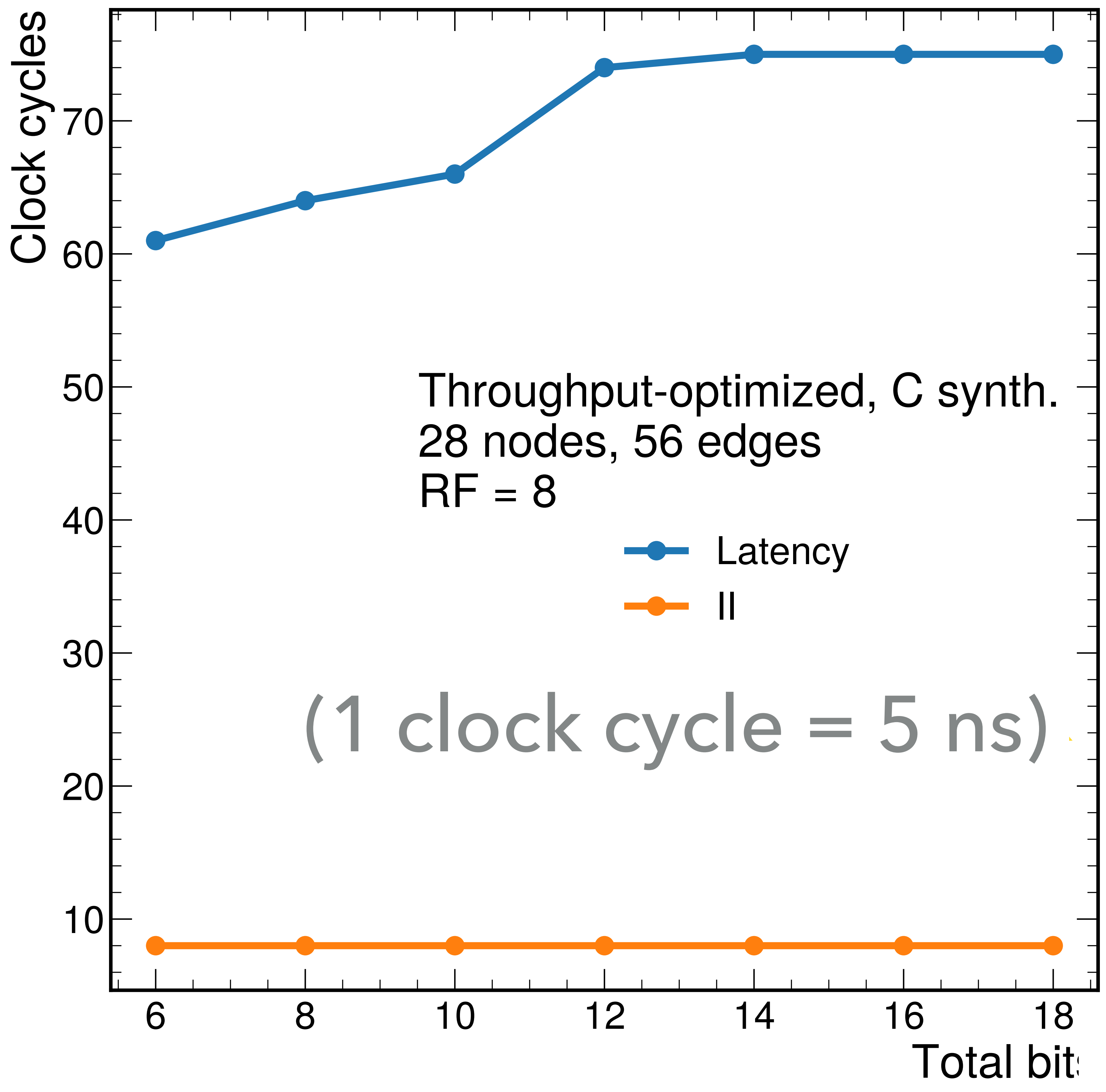




- Small graphs (~30 nodes, ~60 edges) easily fit on 1 FPGA
- Within L1T latency (300 ns) and II (50 ns) requirements

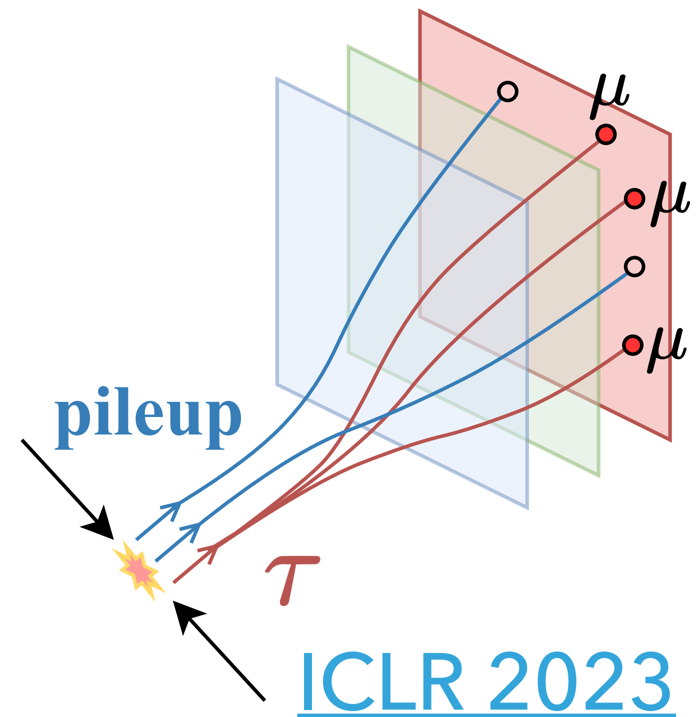
GNN FPGA IMPLEMENTATION



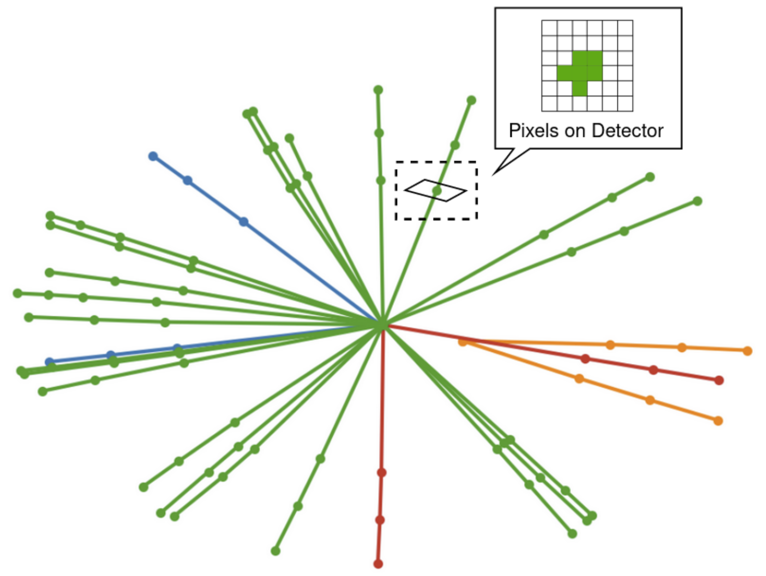


(1 clock cycle = 5 ns)

Similar algorithms for $\tau \to 3\mu$ @ LHC and tracking in sPHENIX



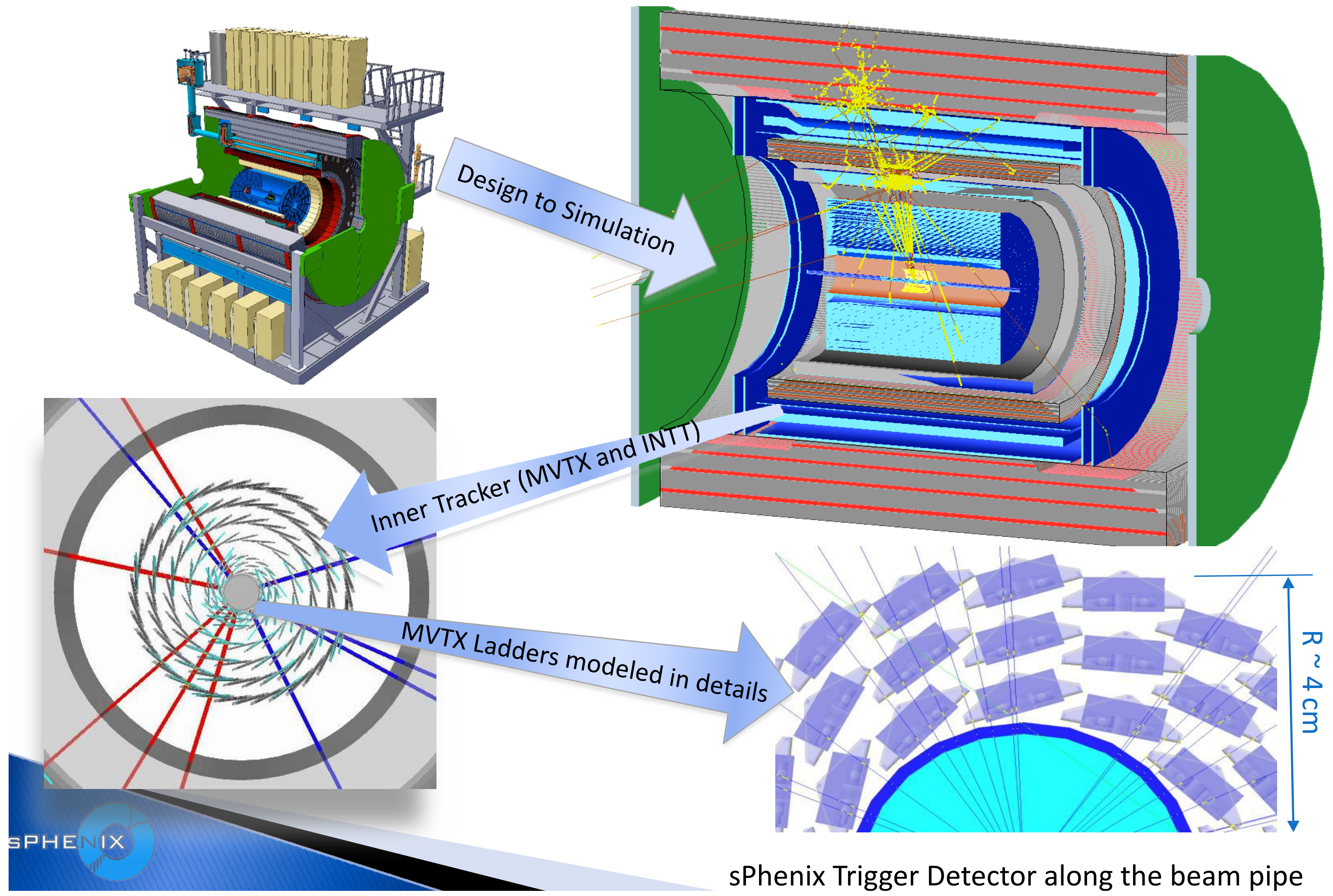
ICLR 2023



ECML PKDD 2022

- Small graphs (~30 nodes, ~60 edges) easily fit on 1 FPGA
- Within L1T latency (300 ns) and II (50 ns) requirements

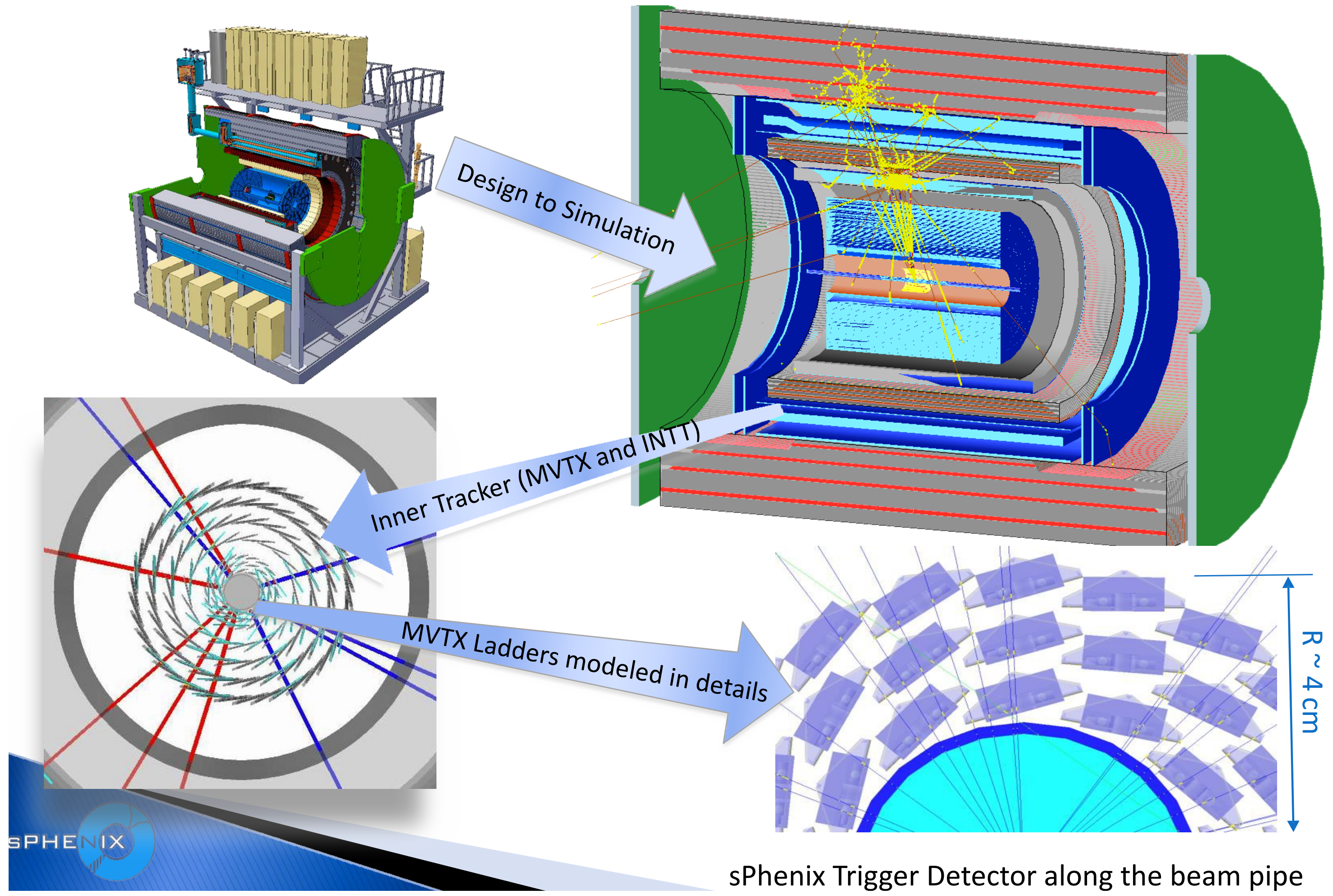
BEYOND LHC



Design to Simulation
Inner Tracker (MVTX and INTT)
MVTX Ladders modeled in details
R ~ 4 cm
sPHENIX
sPhenix Trigger Detector along the beam pipe

BEYOND LHC

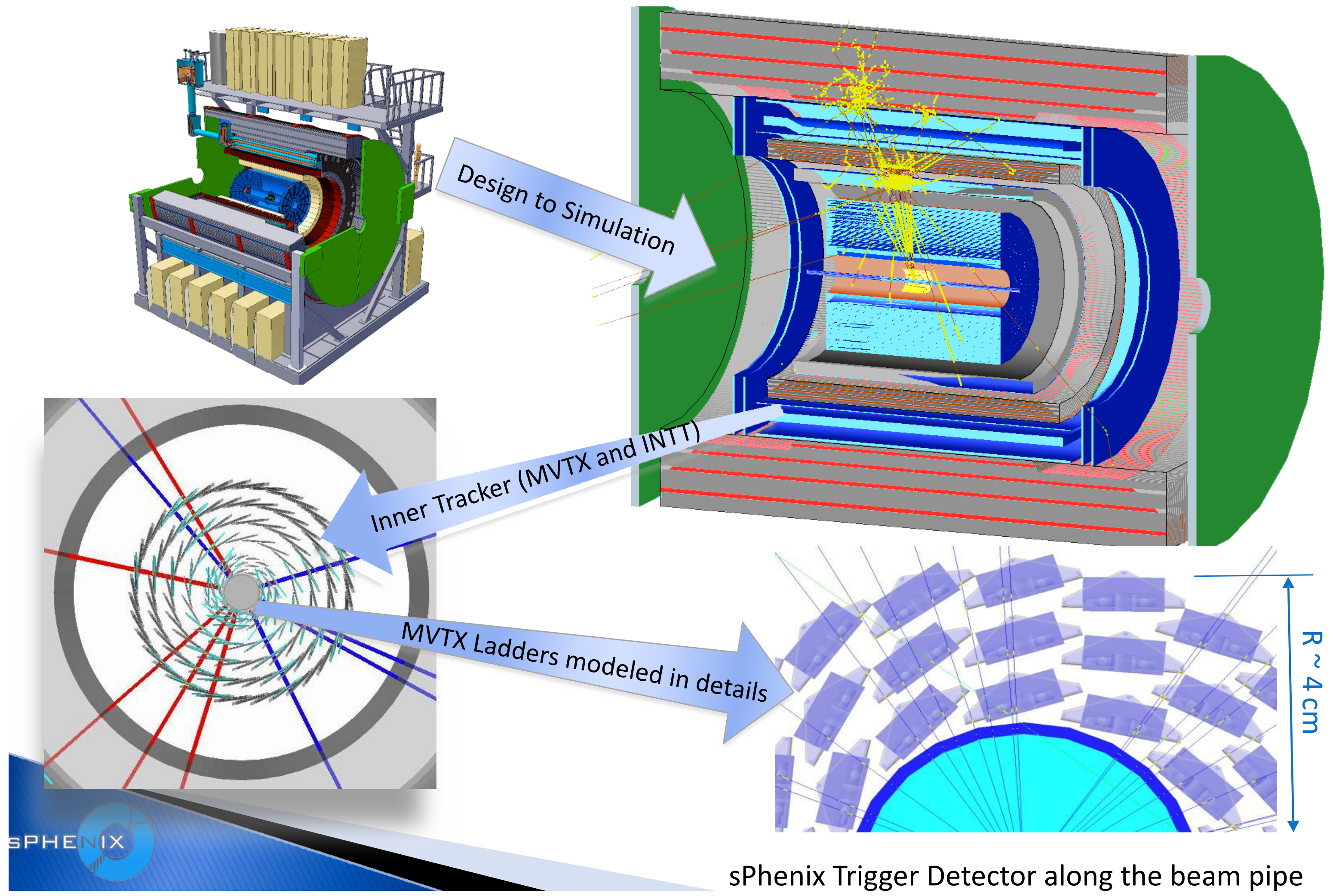
- Nuclear physics

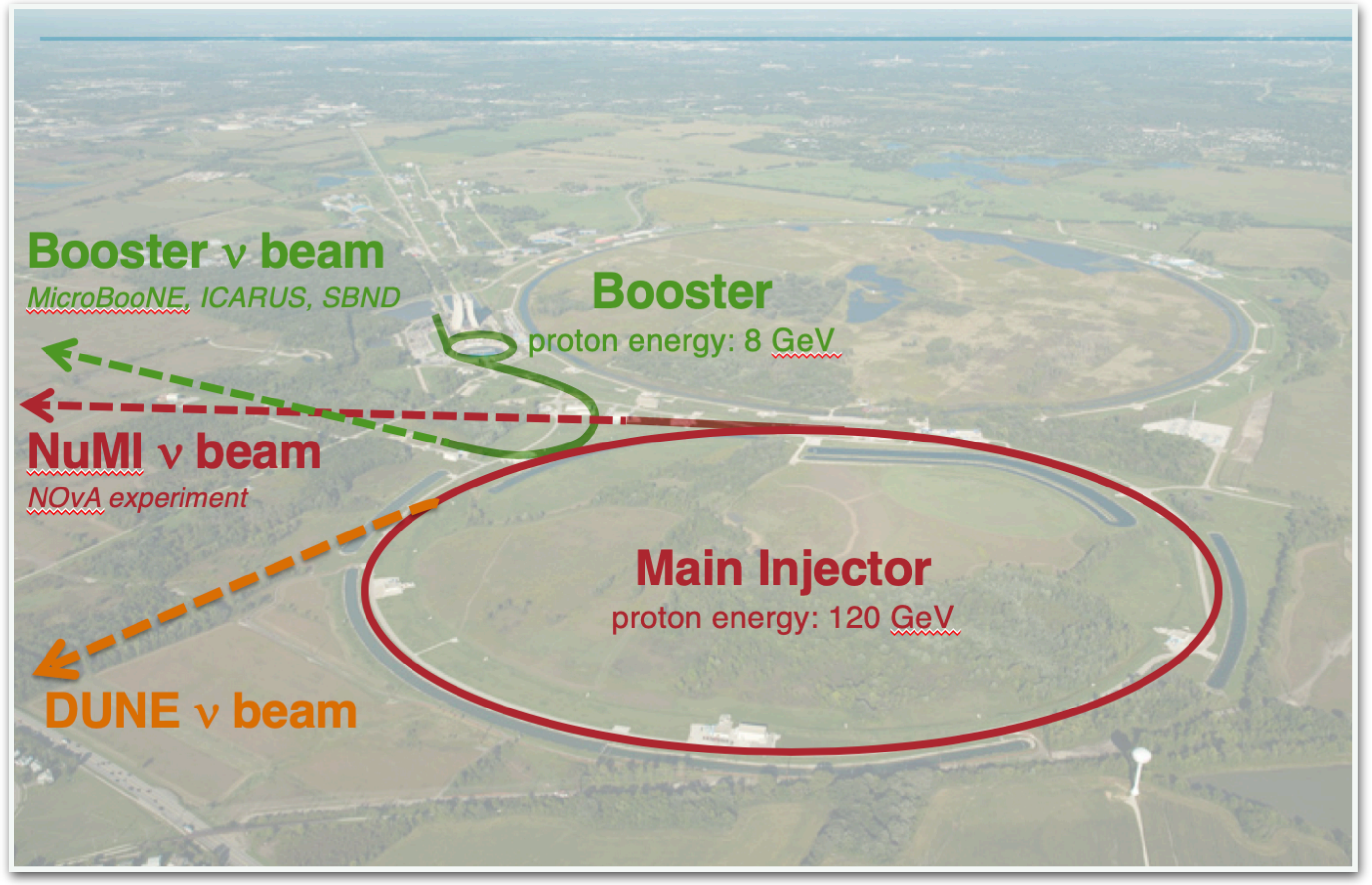


sPhenix Trigger Detector along the beam pipe

BEYOND LHC

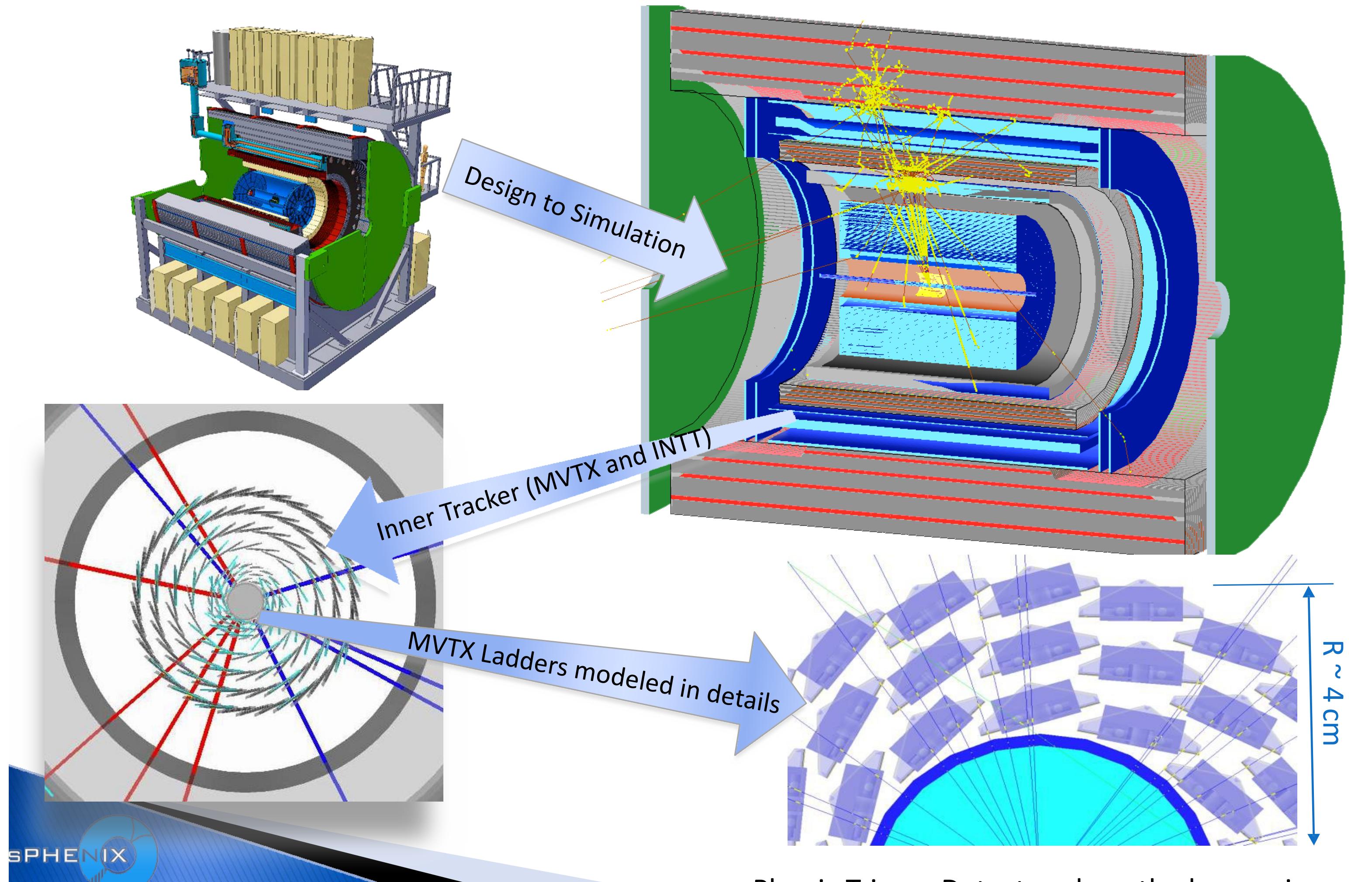
- Nuclear physics
- Accelerator control



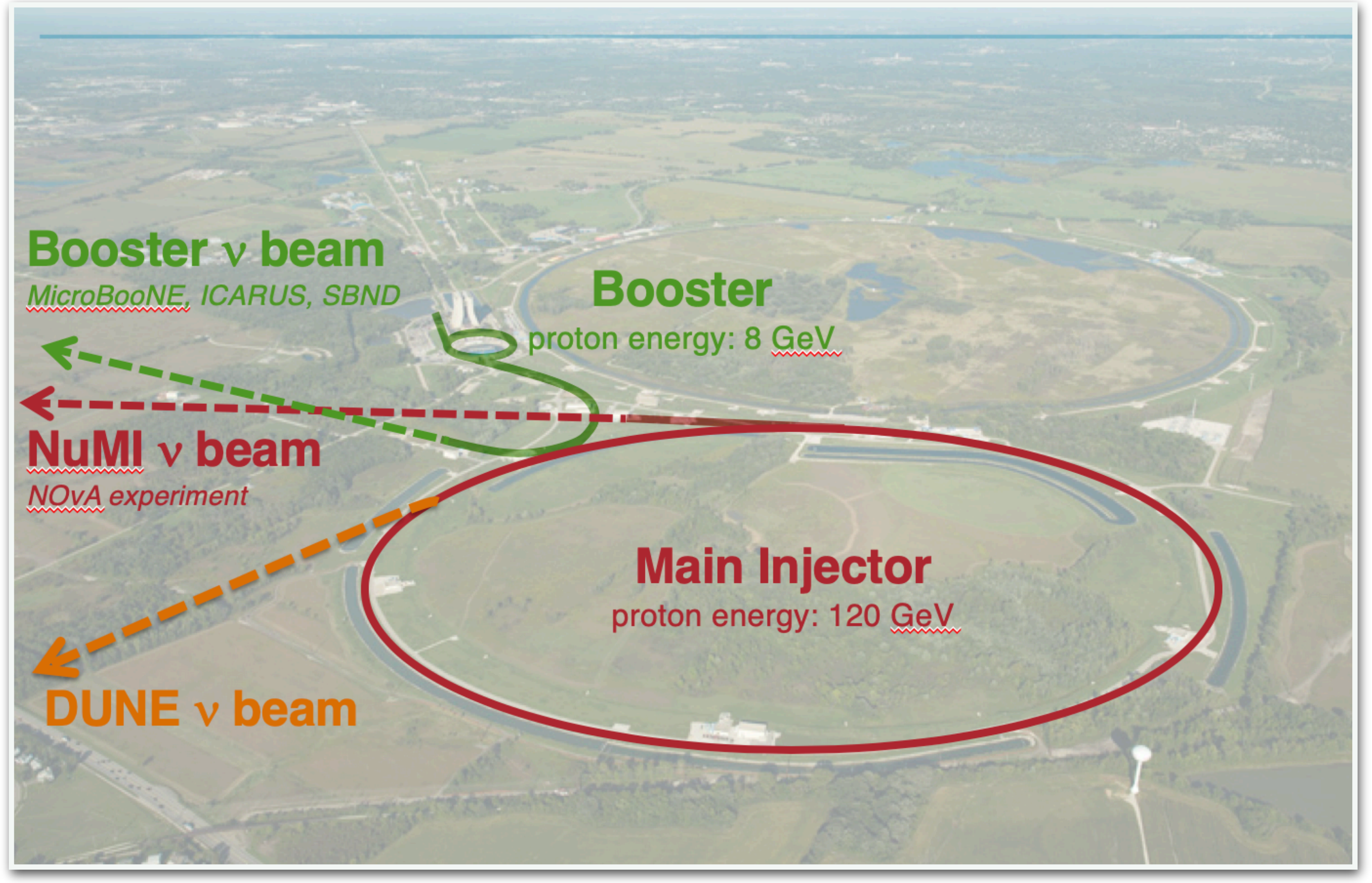


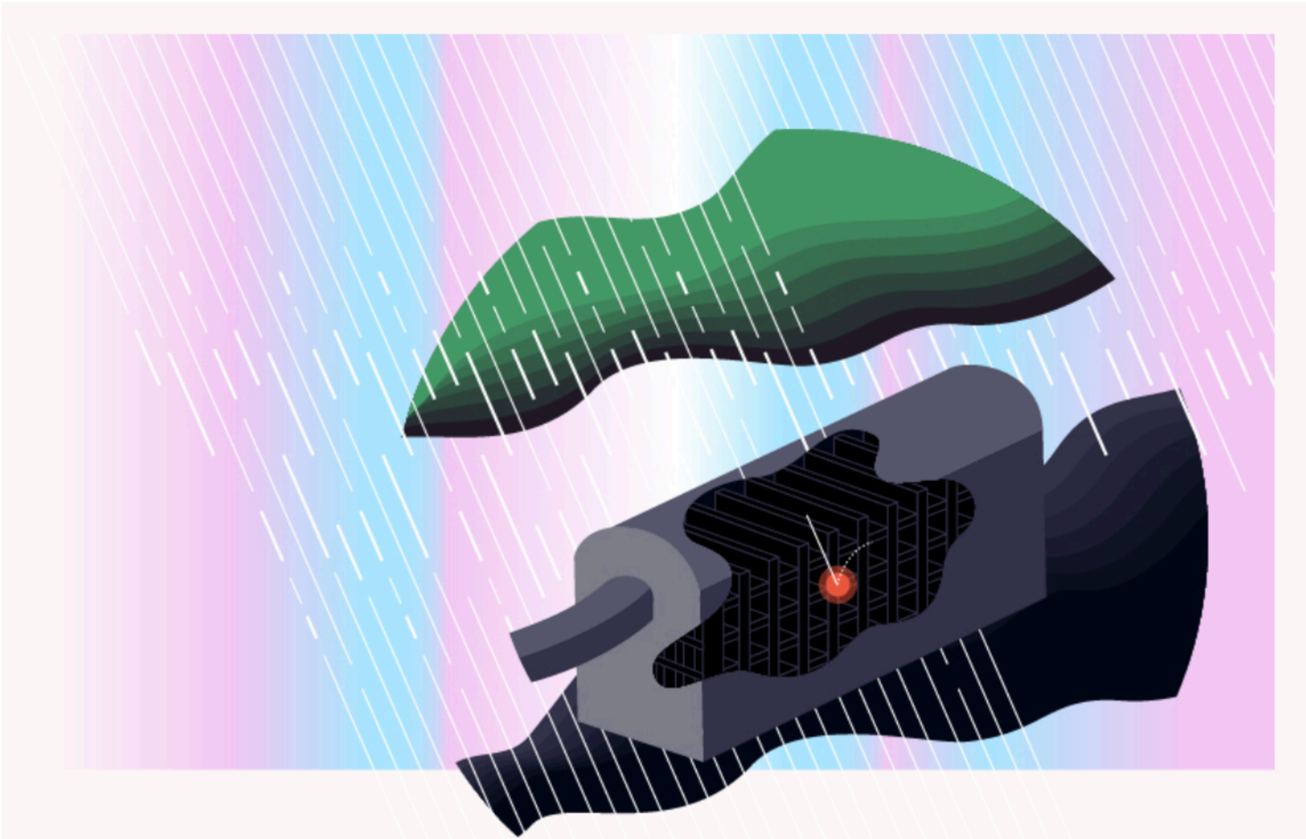
BEYOND LHC

- Nuclear physics
- Accelerator control
- Neutrino physics



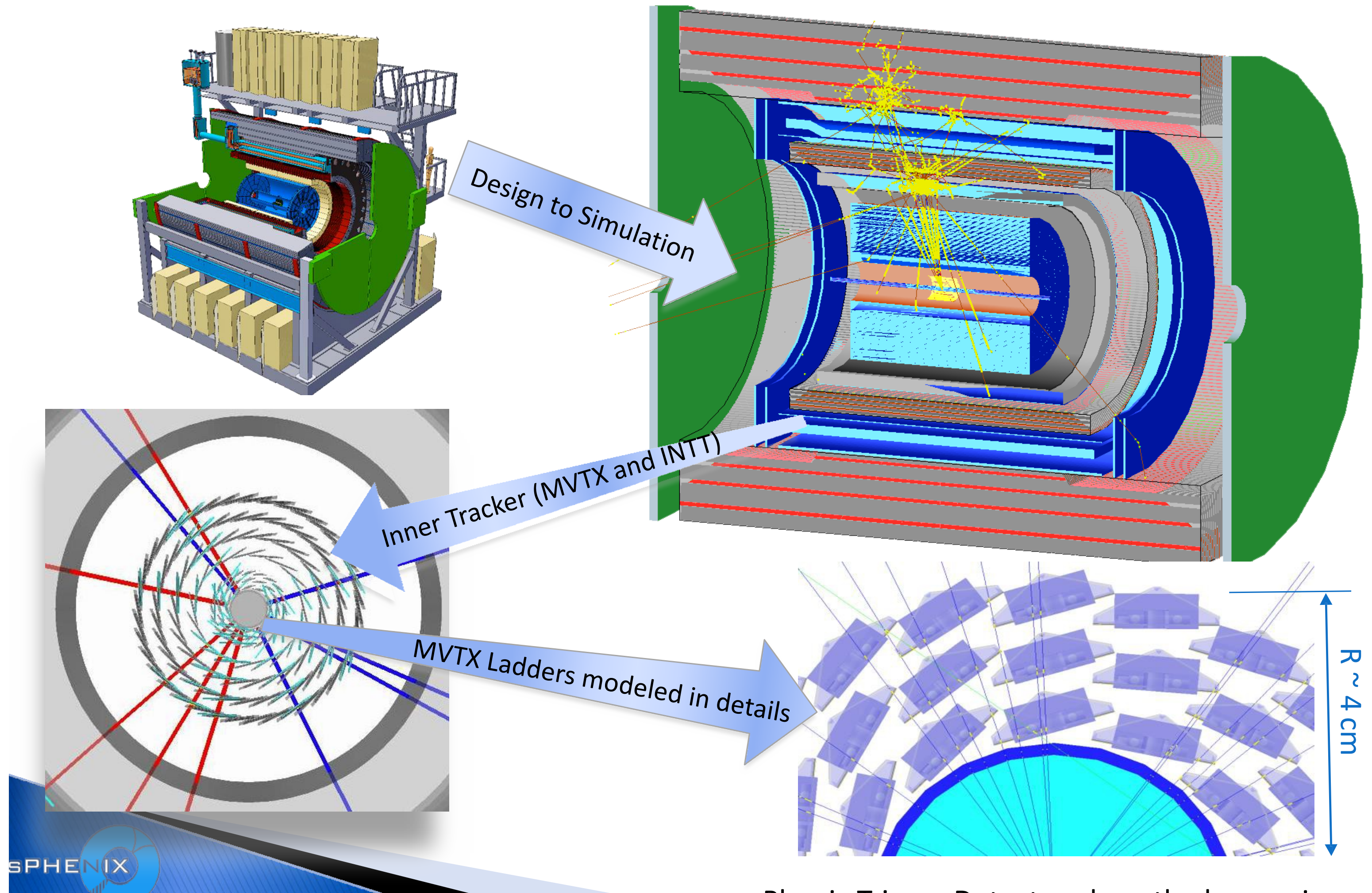
sPhenix Trigger Detector along the beam pipe



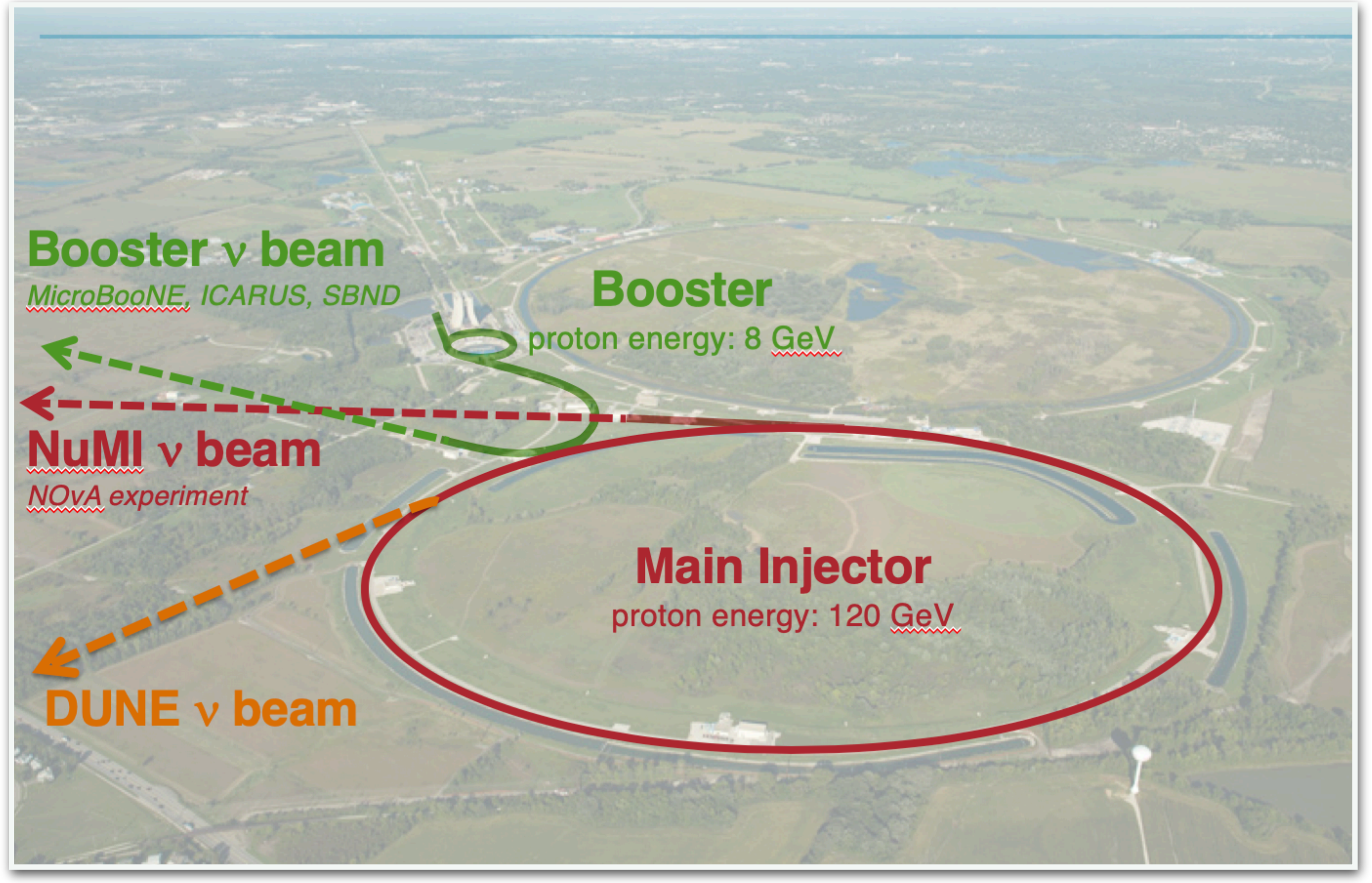


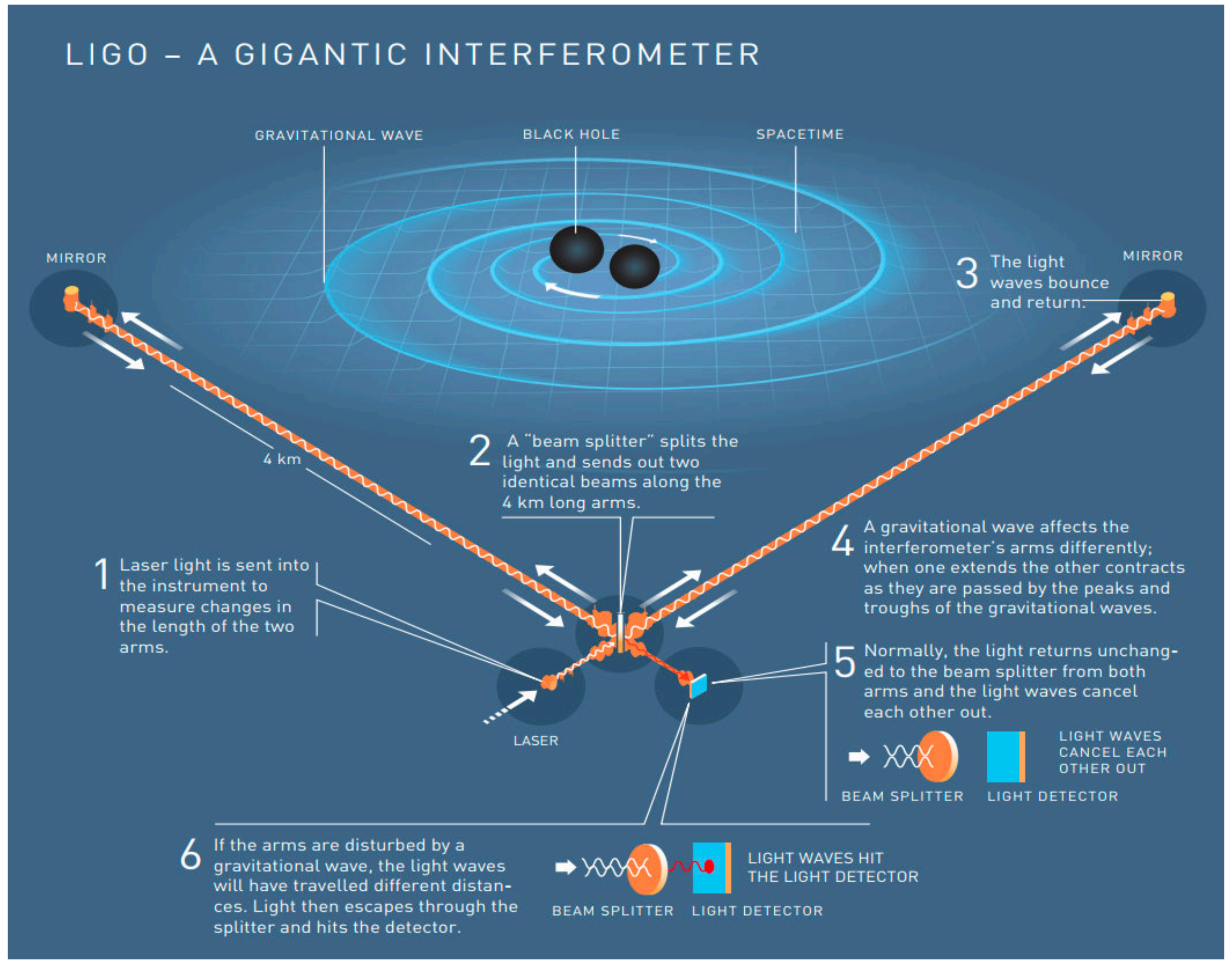
BEYOND LHC

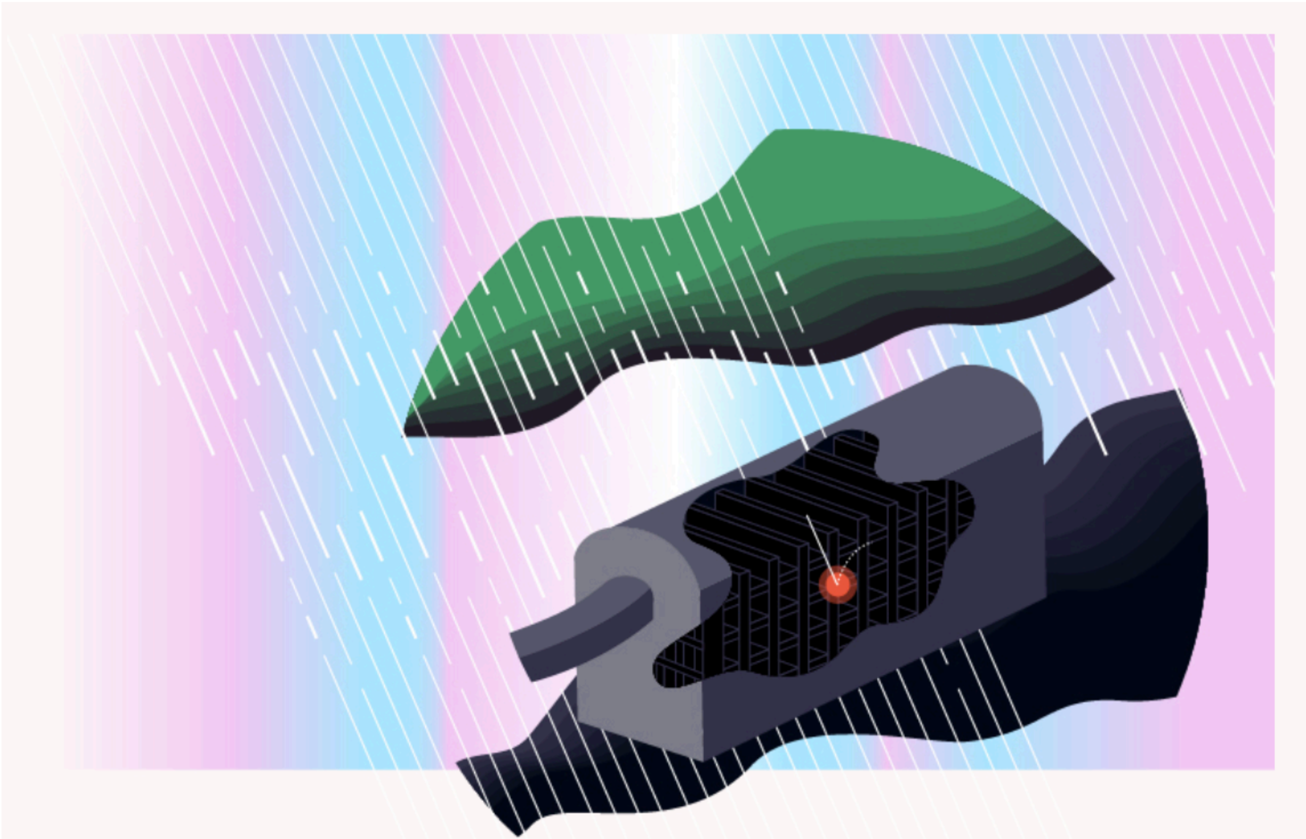
- Nuclear physics
- Accelerator control
- Neutrino physics
- Multi-messenger astronomy



sPhenix Trigger Detector along the beam pipe

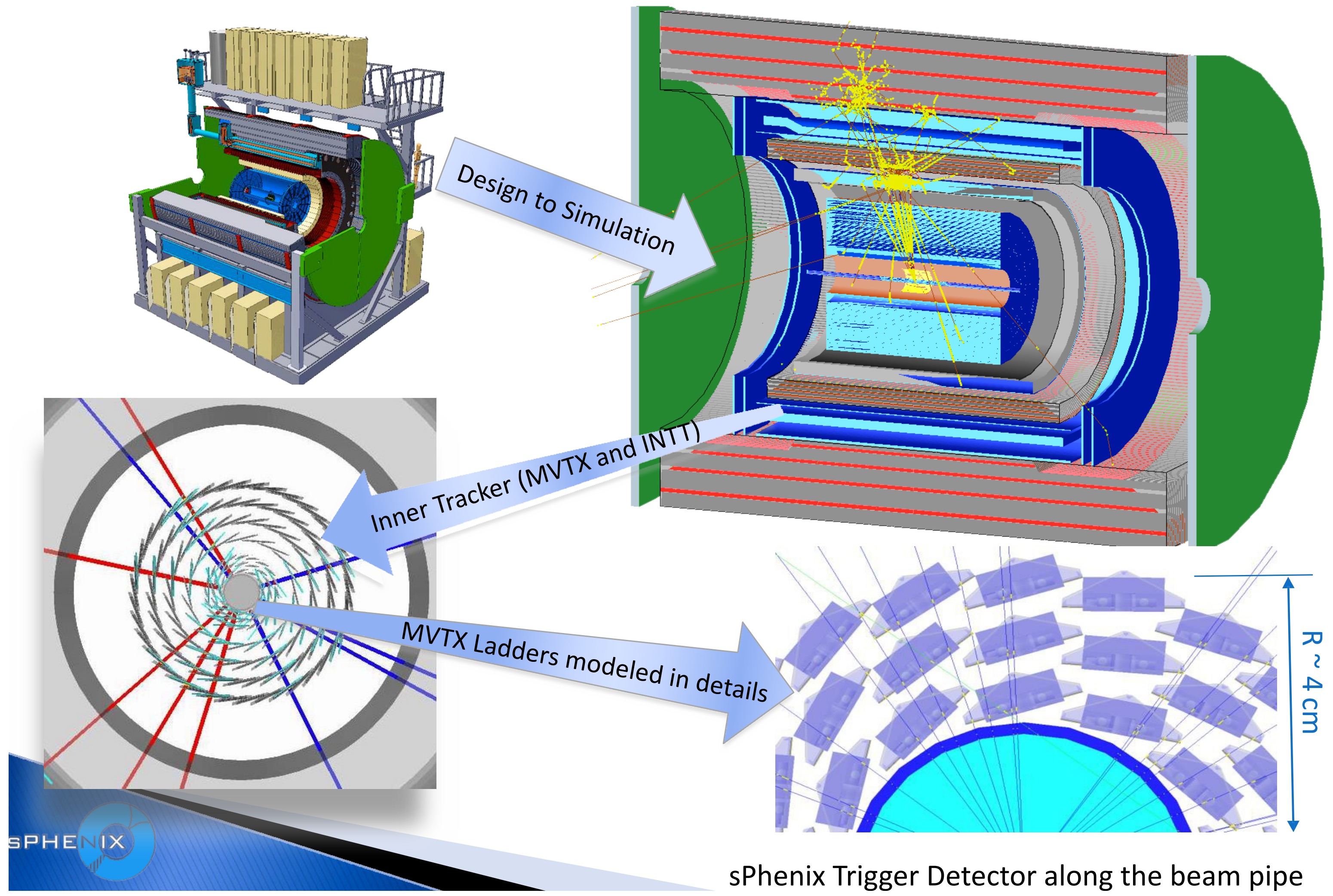




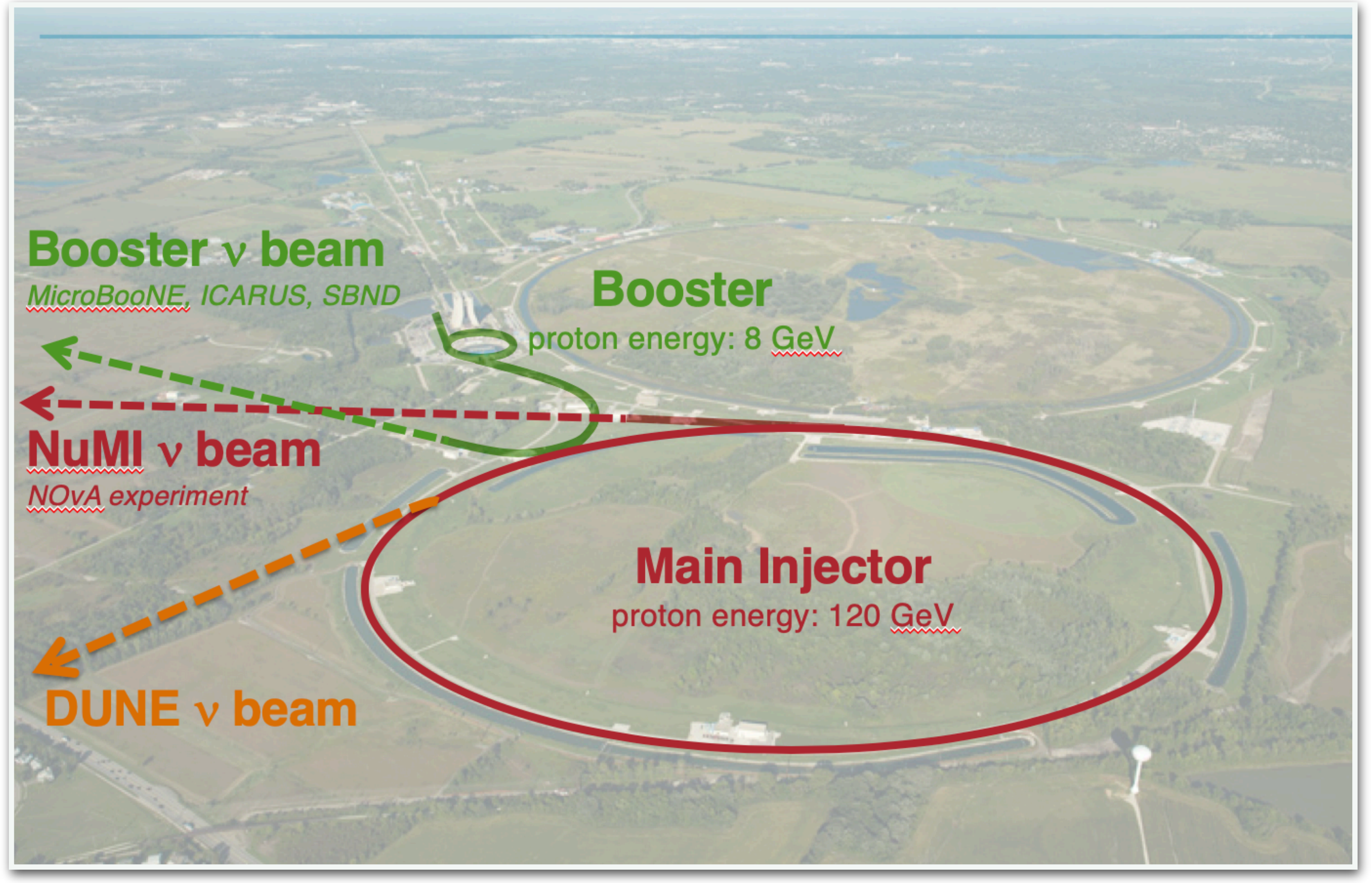


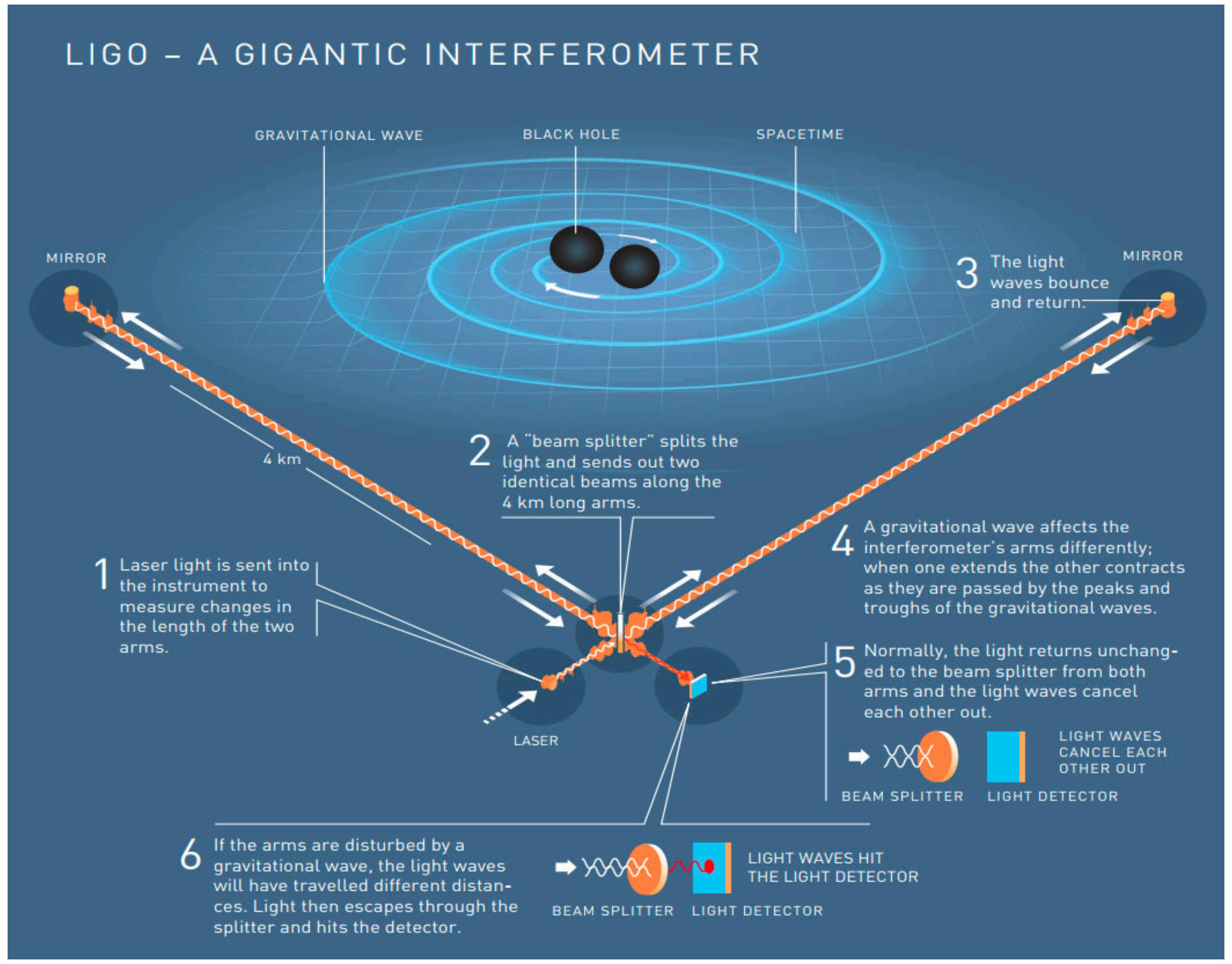
BEYOND LHC

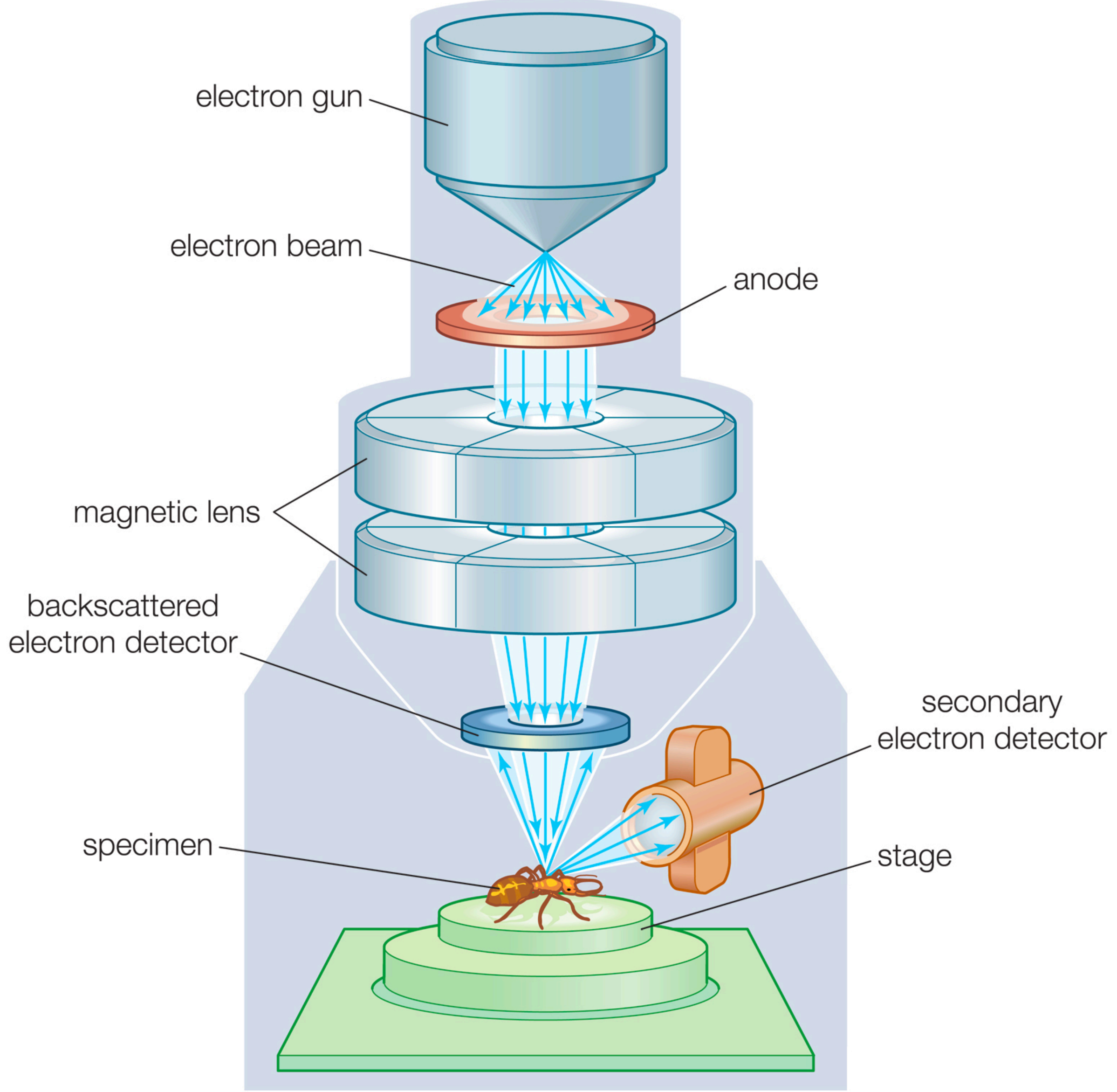
- Nuclear physics
- Accelerator control
- Neutrino physics
- Multi-messenger astronomy
- Electron & X-ray microscopy

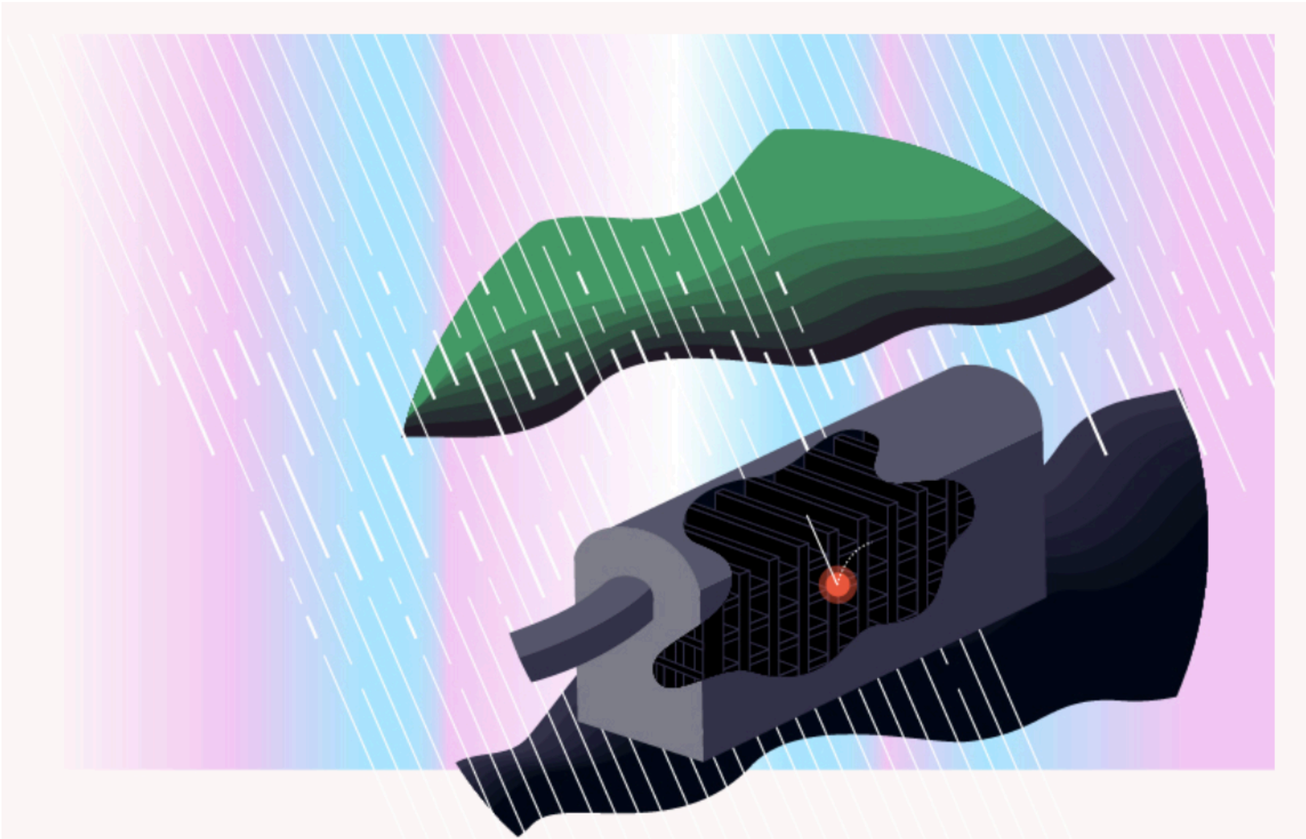


sPhenix Trigger Detector along the beam pipe



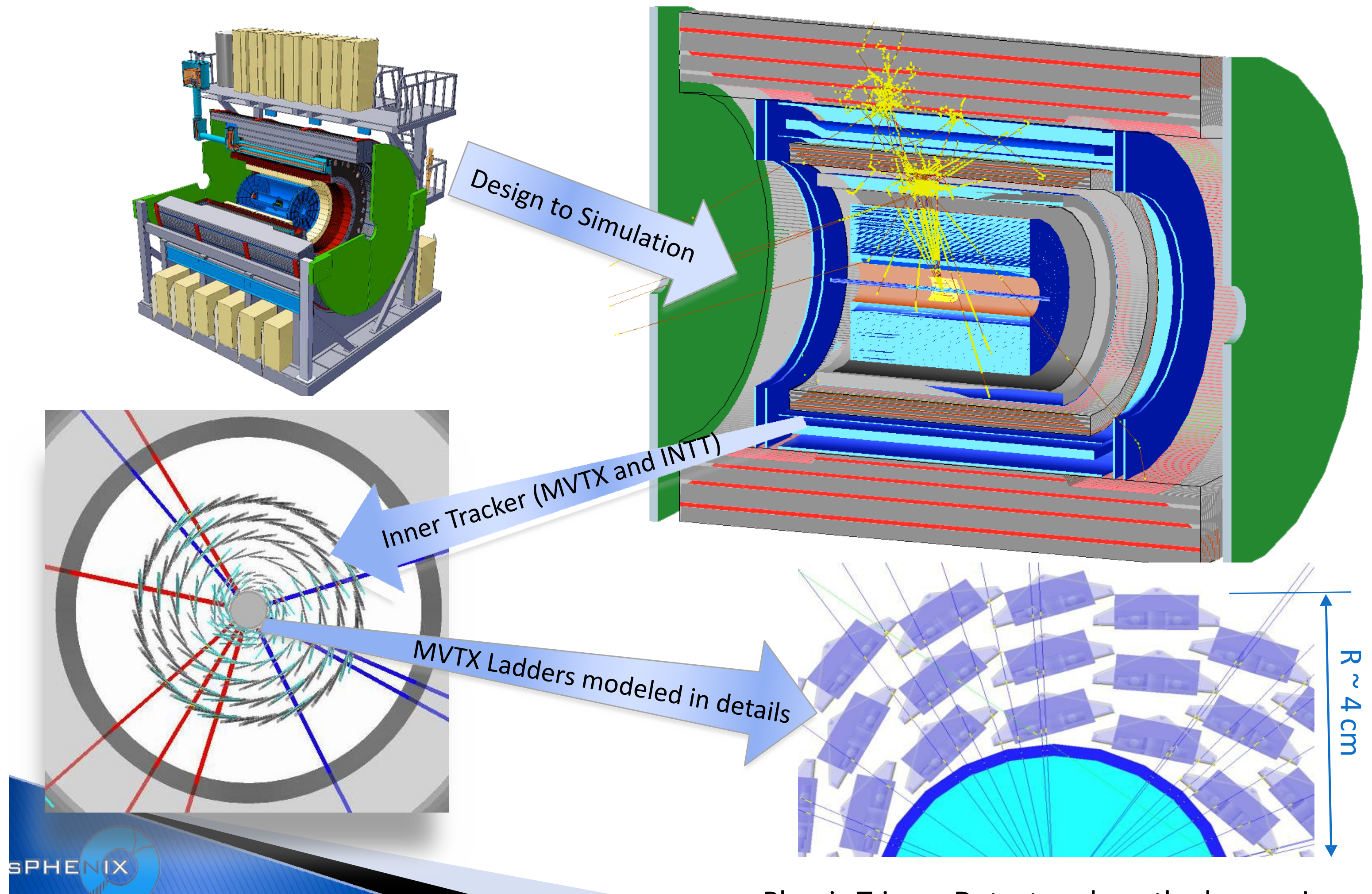




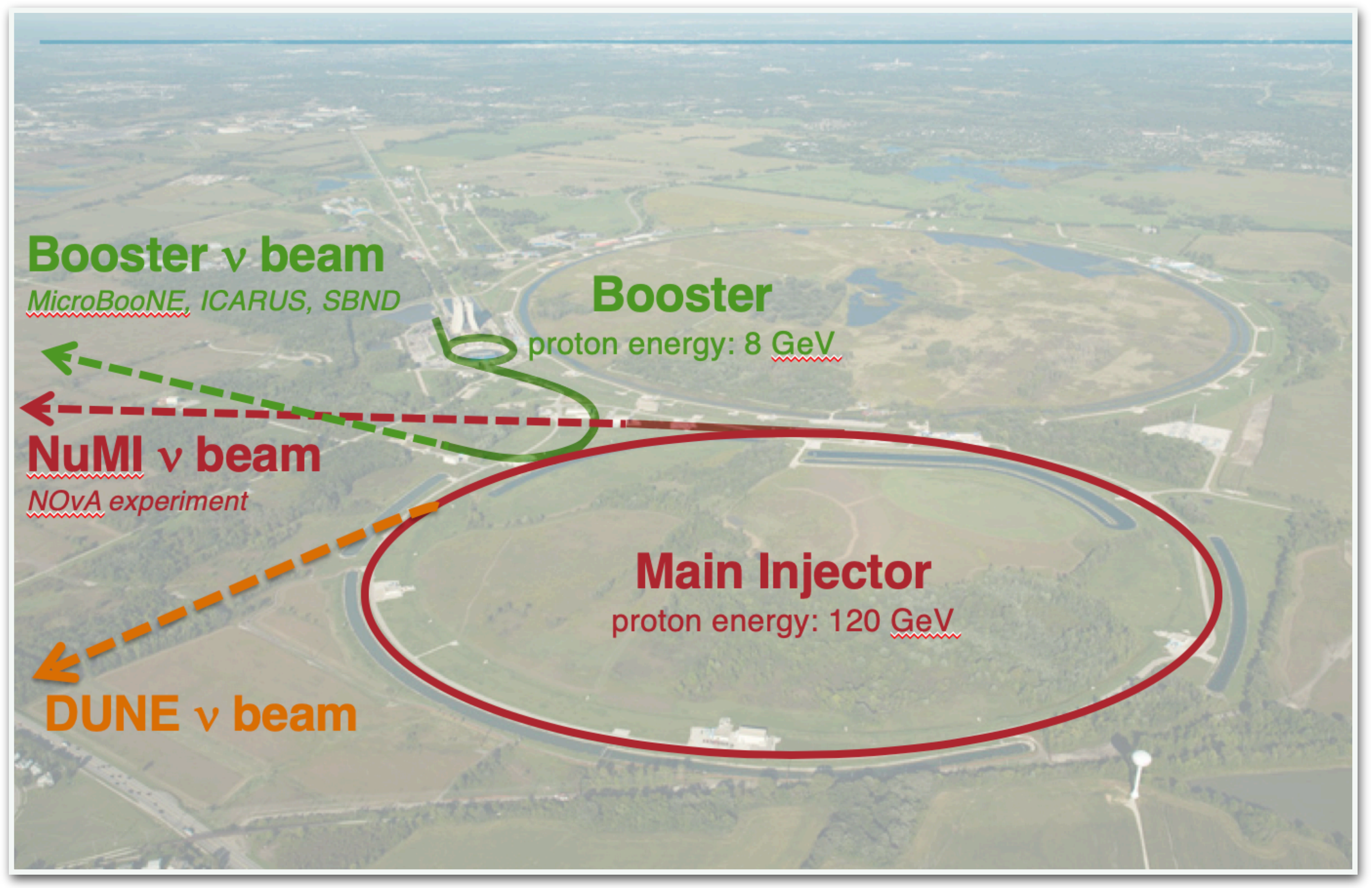


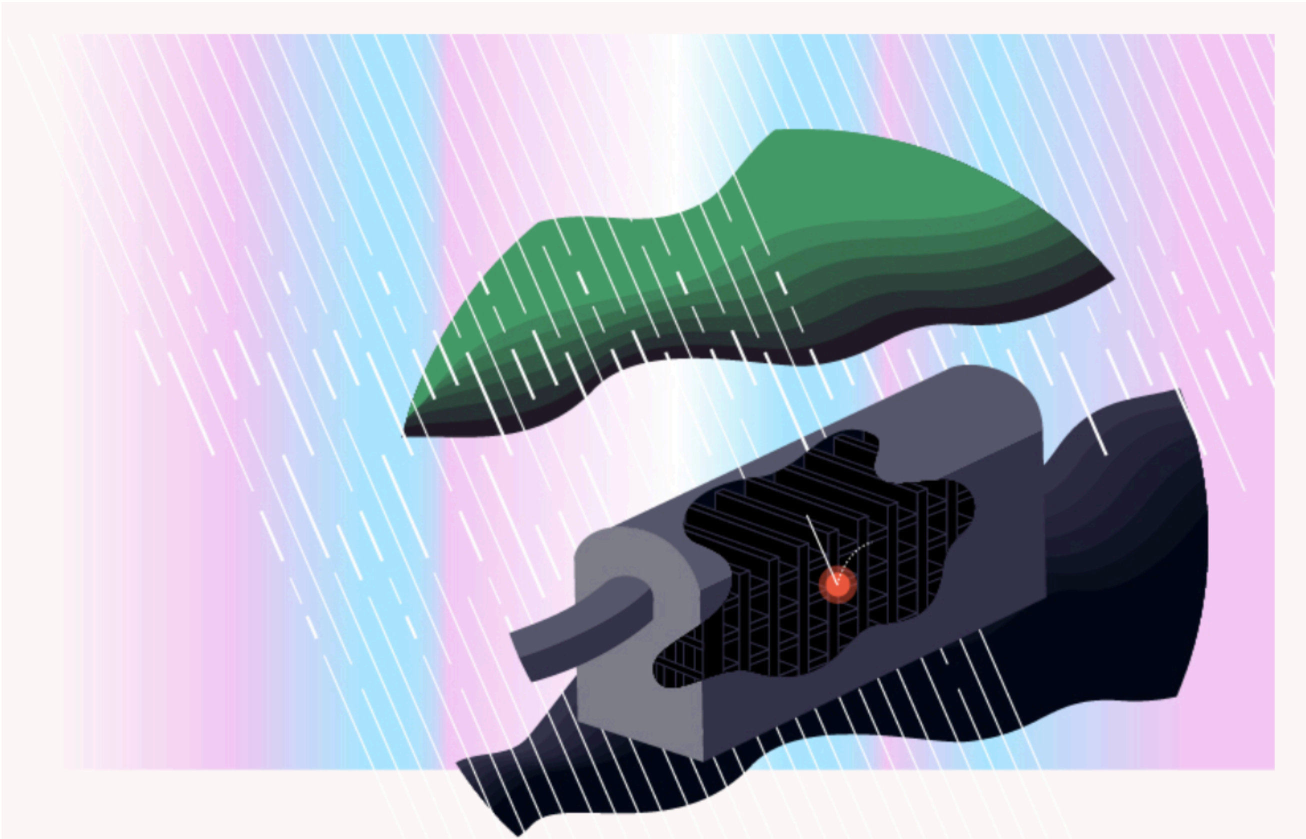
BEYOND LHC

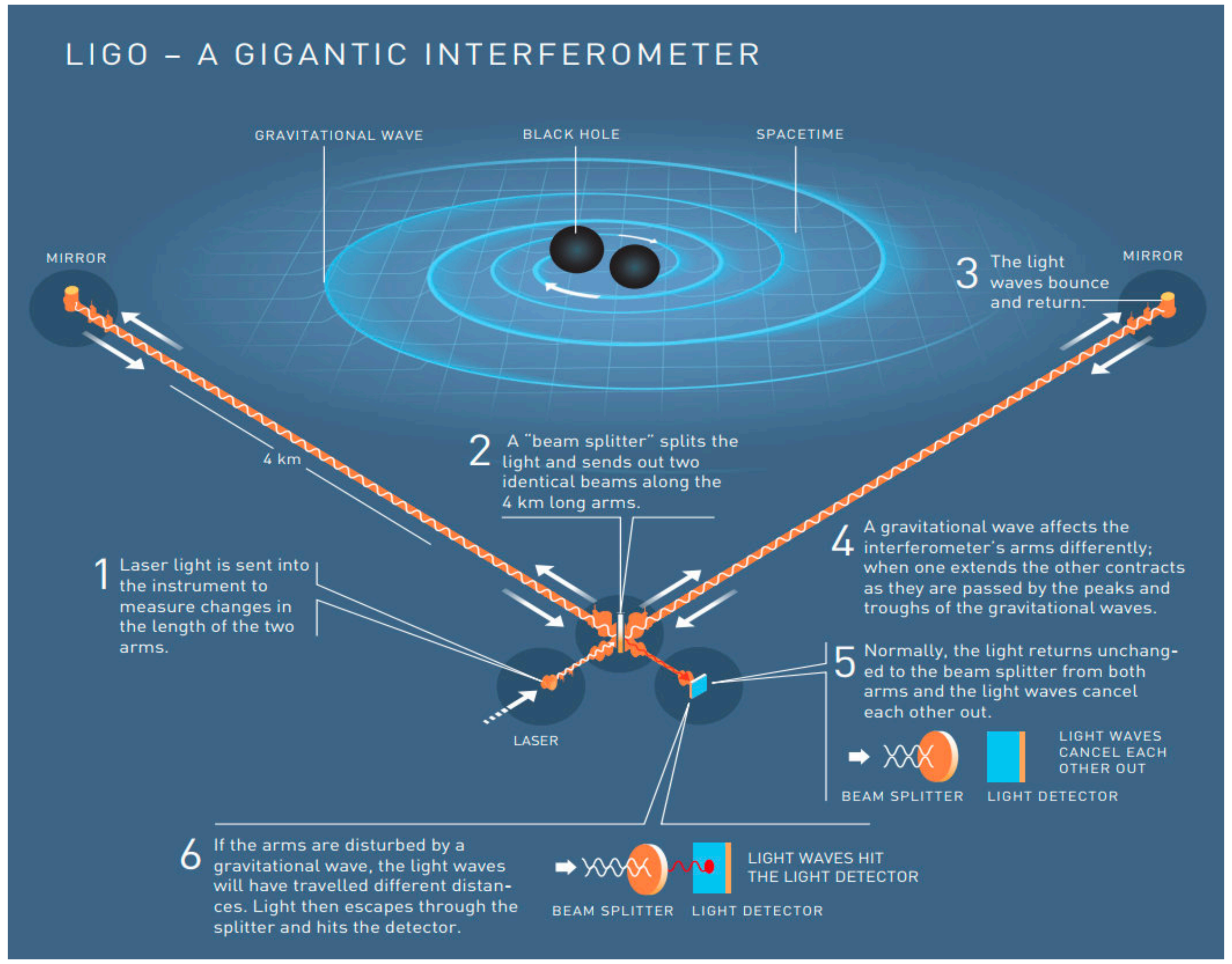
- Nuclear physics
- Accelerator control
- Neutrino physics
- Multi-messenger astronomy
- Electron & X-ray microscopy
- Neuroscience

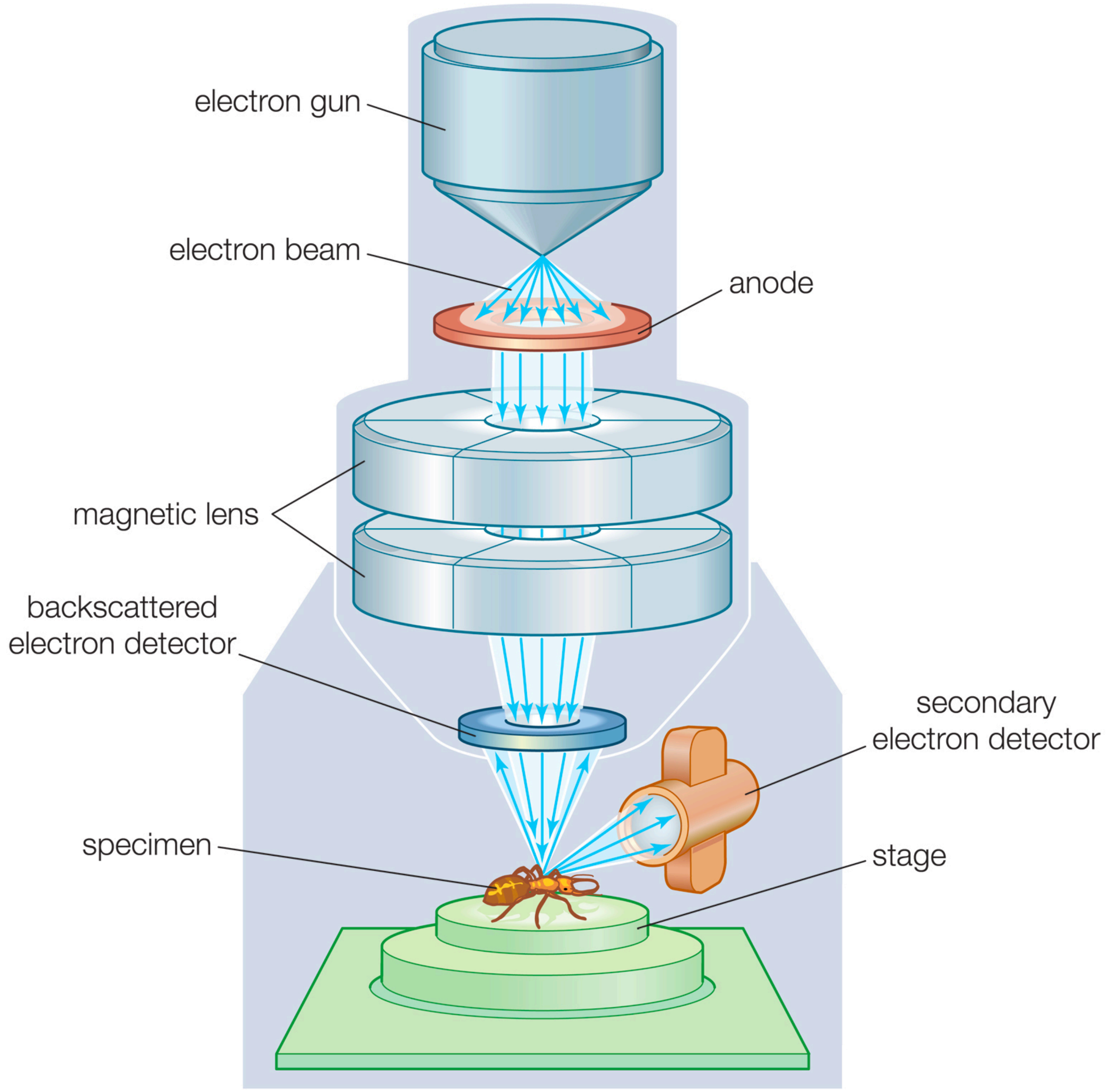


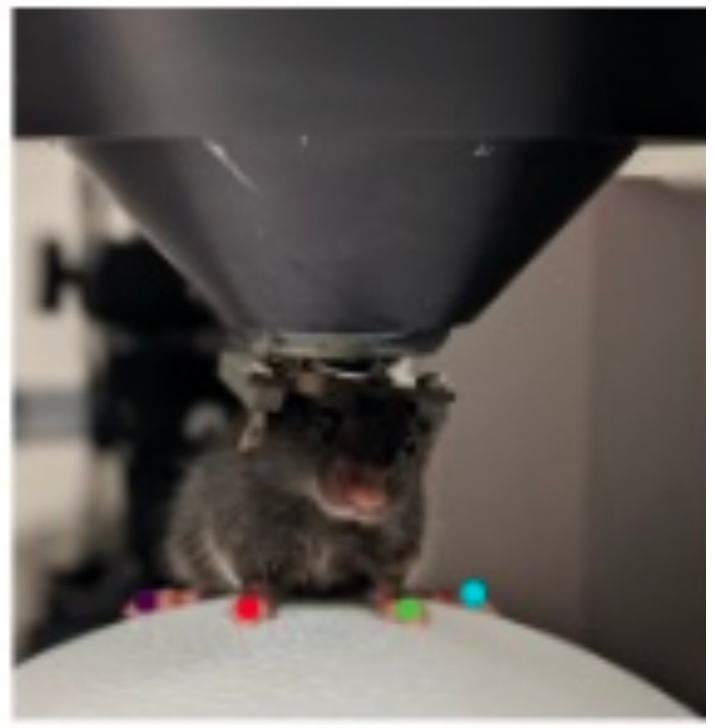
sPhenix Trigger Detector along the beam pipe

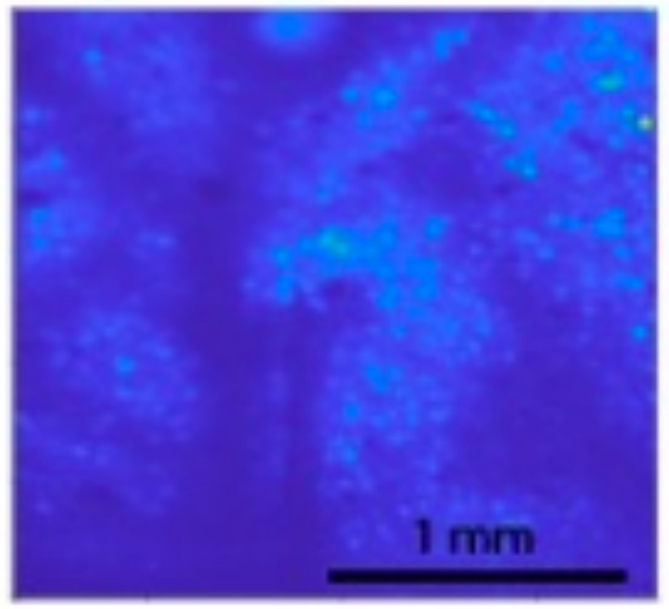












SUMMARY & OUTLOOK

- ML allows us to better reconstruct our data and save potentially overlooked data

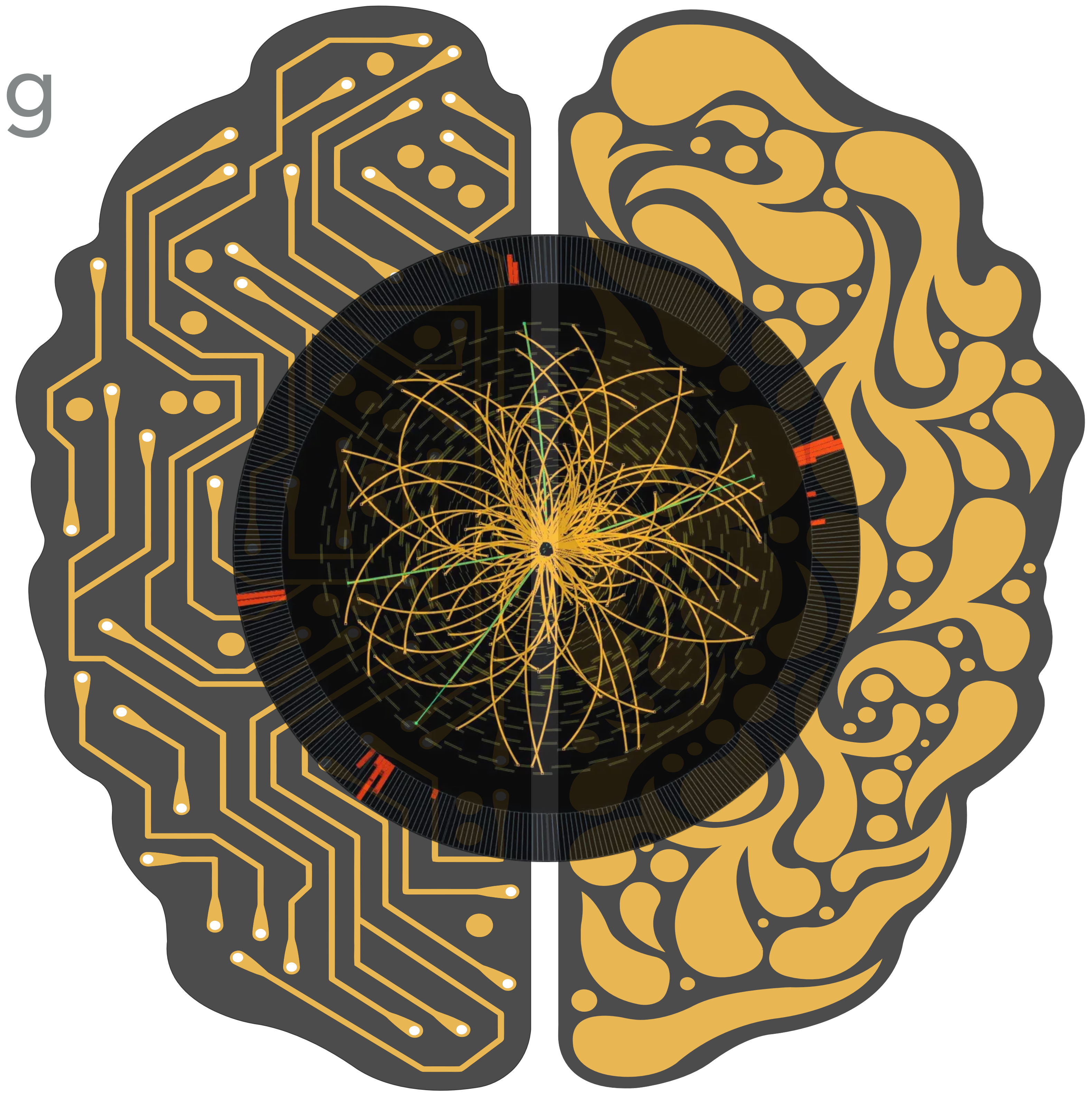
SUMMARY & OUTLOOK

- ML allows us to better reconstruct our data and save potentially overlooked data
- ***Codesign*** principles can enable ML on hardware with stringent constraints

SUMMARY & OUTLOOK

- ML allows us to better reconstruct our data and save potentially overlooked data
- ***Codesign*** principles can enable ML on hardware with stringent constraints
- Community (fastmachinelearning.org, e-group hls-fml@cern.ch) and Institute (a3d3.ai) developing open-source tools and techniques to enable this
  - hls4ml: expanding open-source toolkit for translating ML into hardware aimed at trigger applications and more…

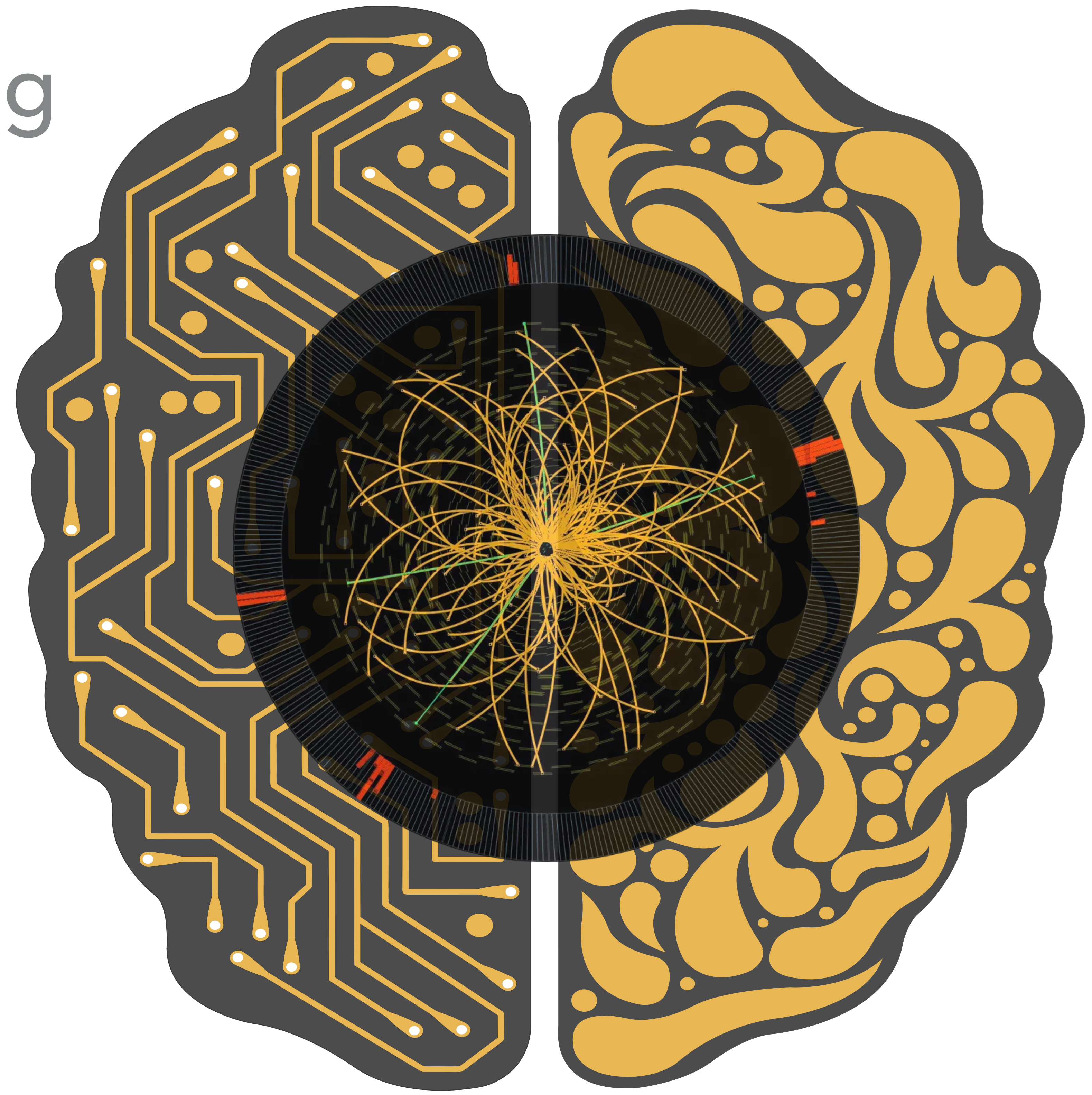
Accelerated AI Algorithms for Data-Driven Discovery



SUMMARY & OUTLOOK

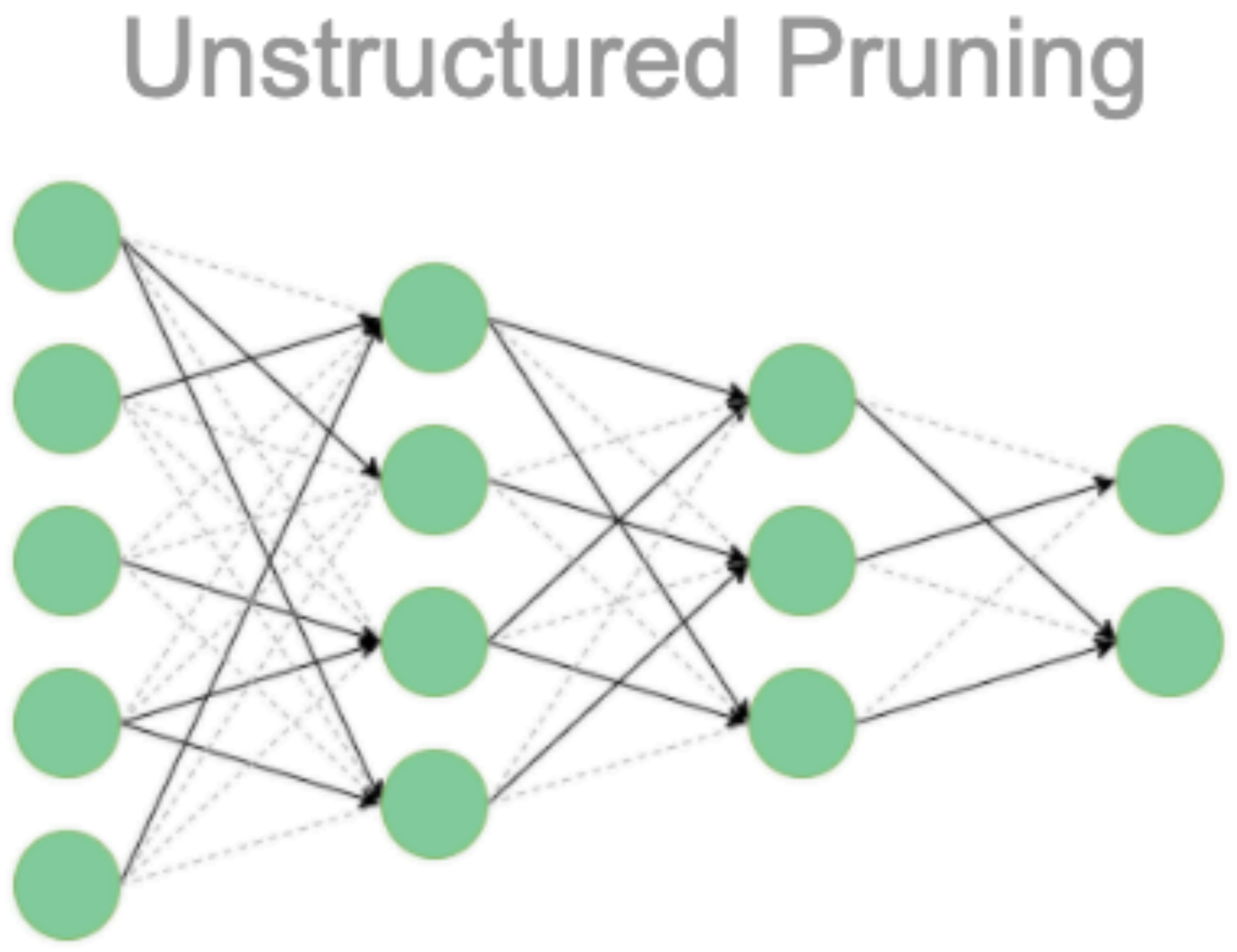
- ML allows us to better reconstruct our data and save potentially overlooked data
- ***Codesign*** principles can enable ML on hardware with stringent constraints
- Community (fastmachinelearning.org, e-group hls-fml@cern.ch) and Institute (a3d3.ai) developing open-source tools and techniques to enable this
  - hls4ml: expanding open-source toolkit for translating ML into hardware aimed at trigger applications and more…
- Applications range from momentum regression, to b-tagging, tracking, and more!
  - Enhance **future particle physics program**

Accelerated AI Algorithms for Data-Driven Discovery



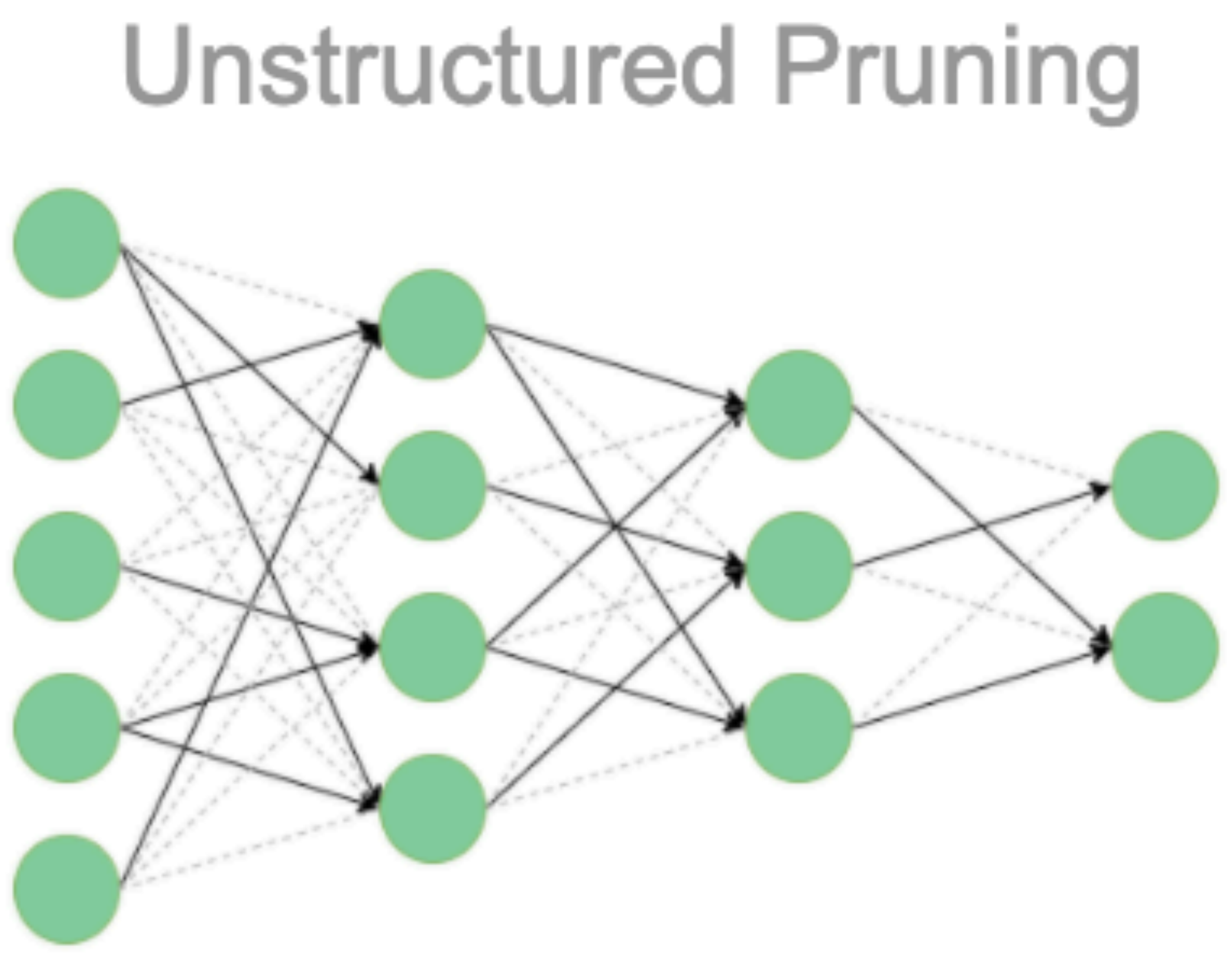
BACKUP

STRUCTURED VS UNSTRUCTURED PRUNING



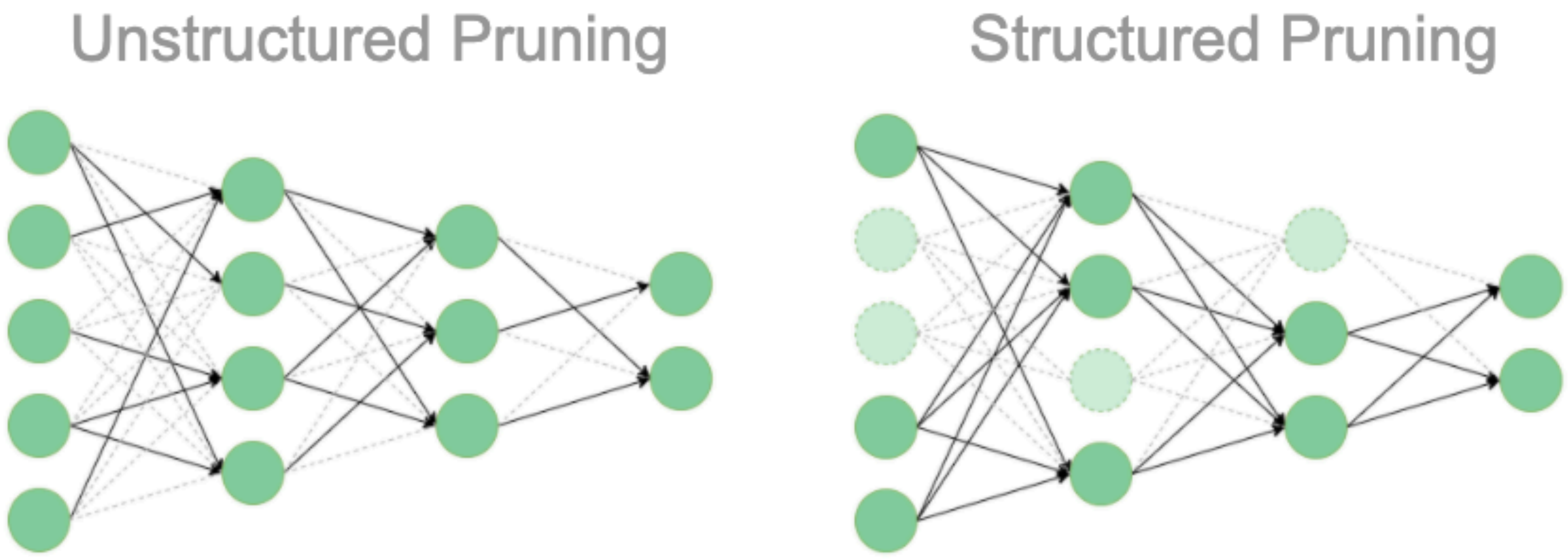
STRUCTURED VS UNSTRUCTURED PRUNING

- Unstructured pruning: removing some connections regardless of placement

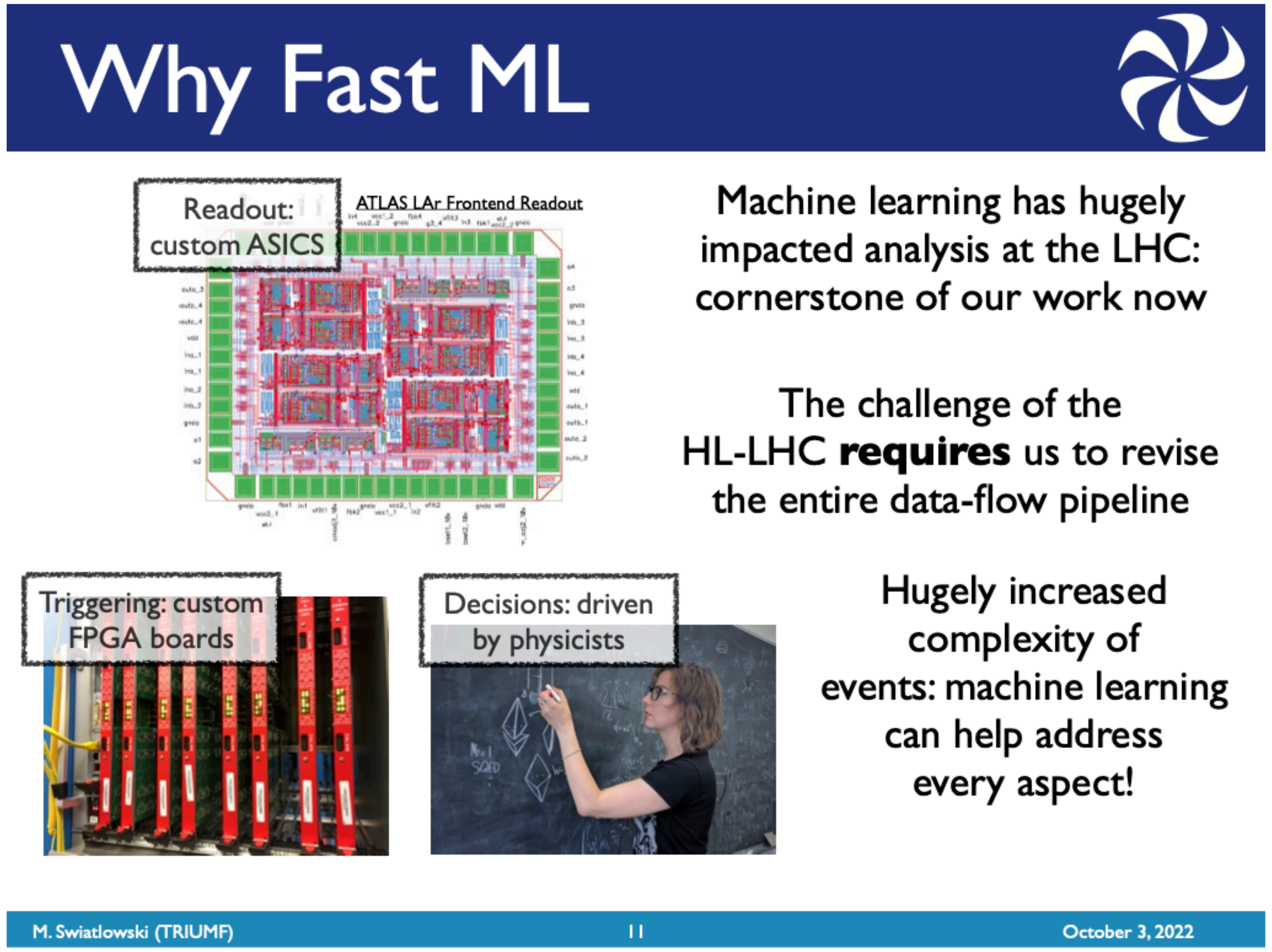


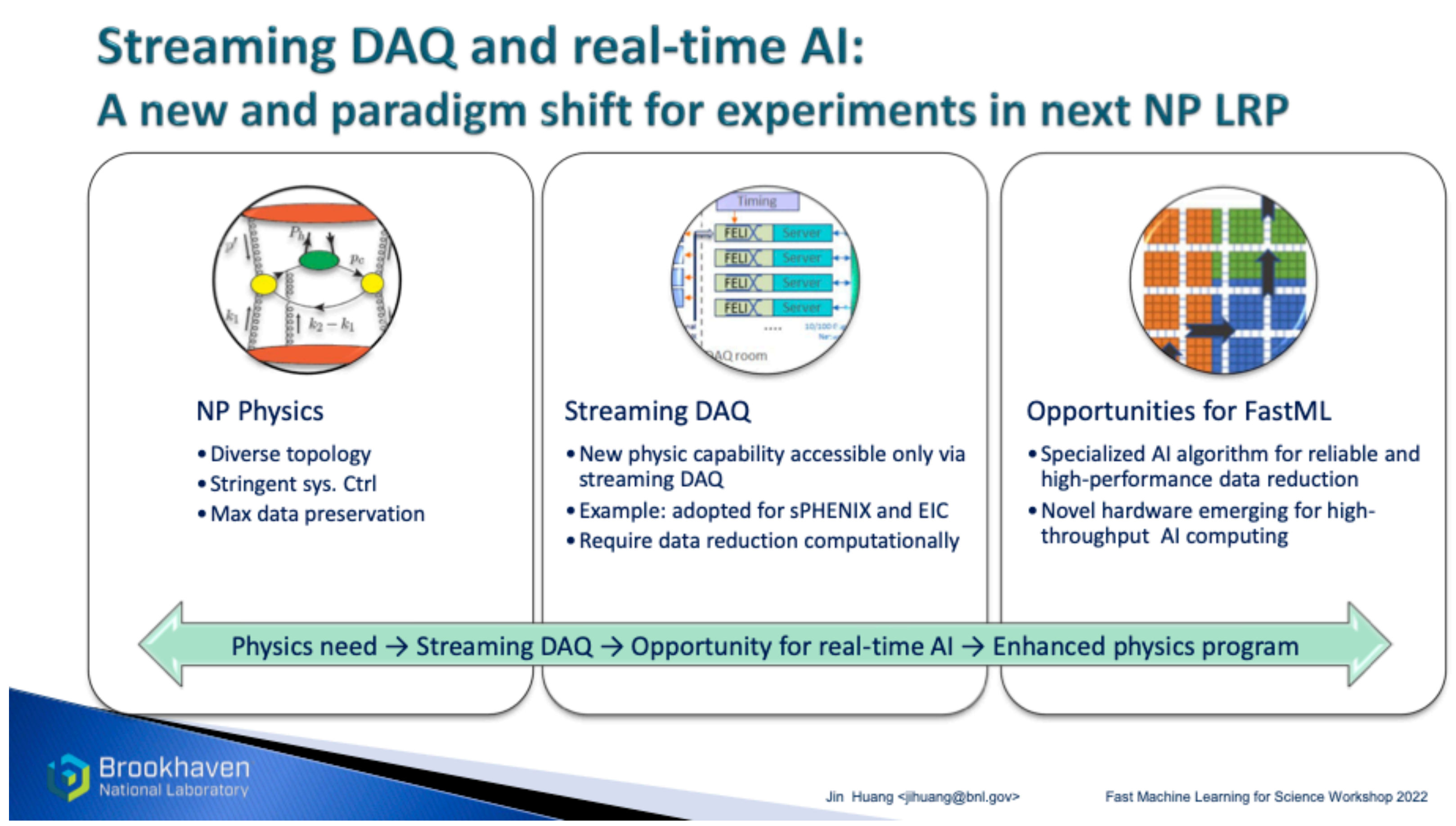
STRUCTURED VS UNSTRUCTURED PRUNING

- Unstructured pruning: removing some connections regardless of placement
- Structured pruning: removing all input/output connections of particular nodes

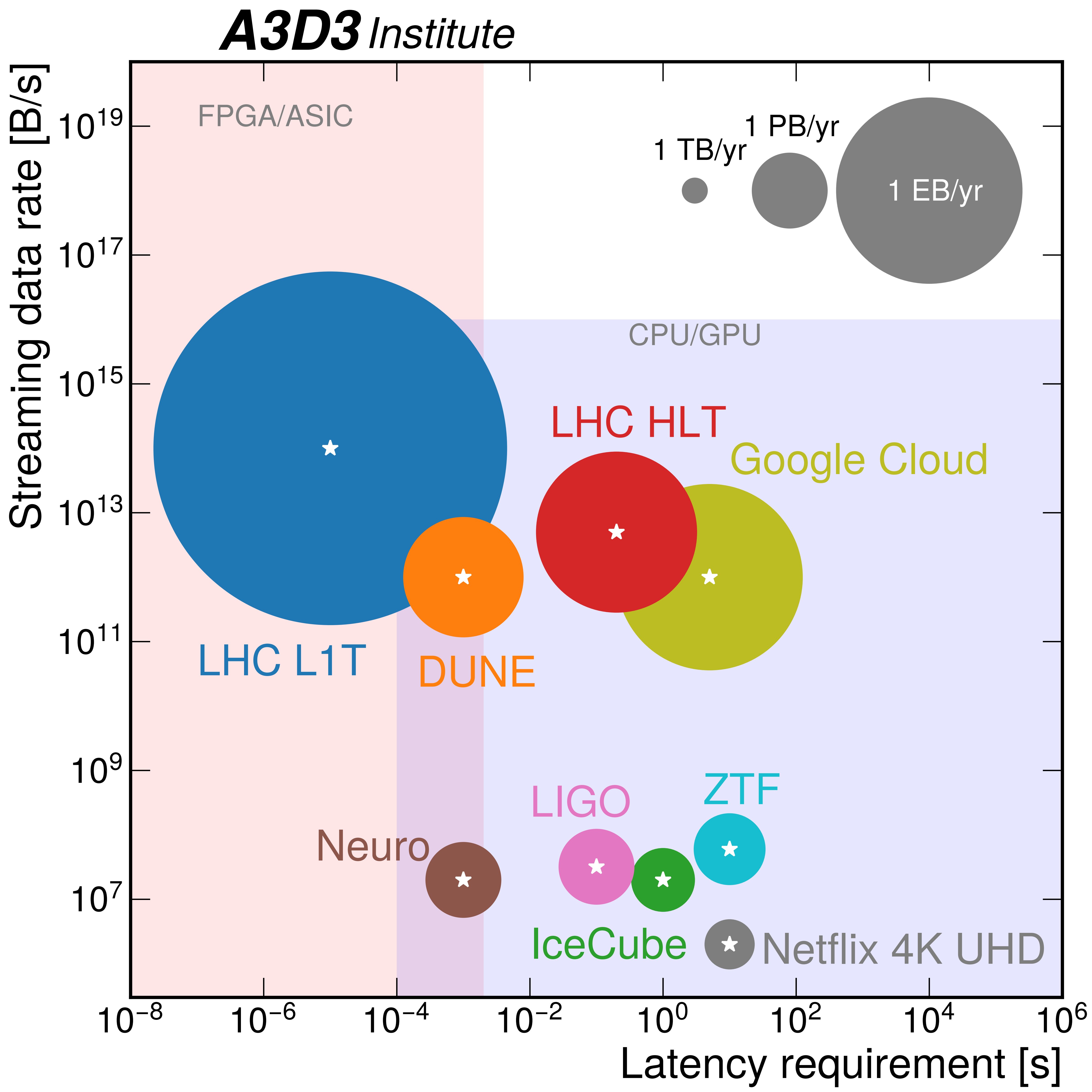


- Excellent overview talks for reference

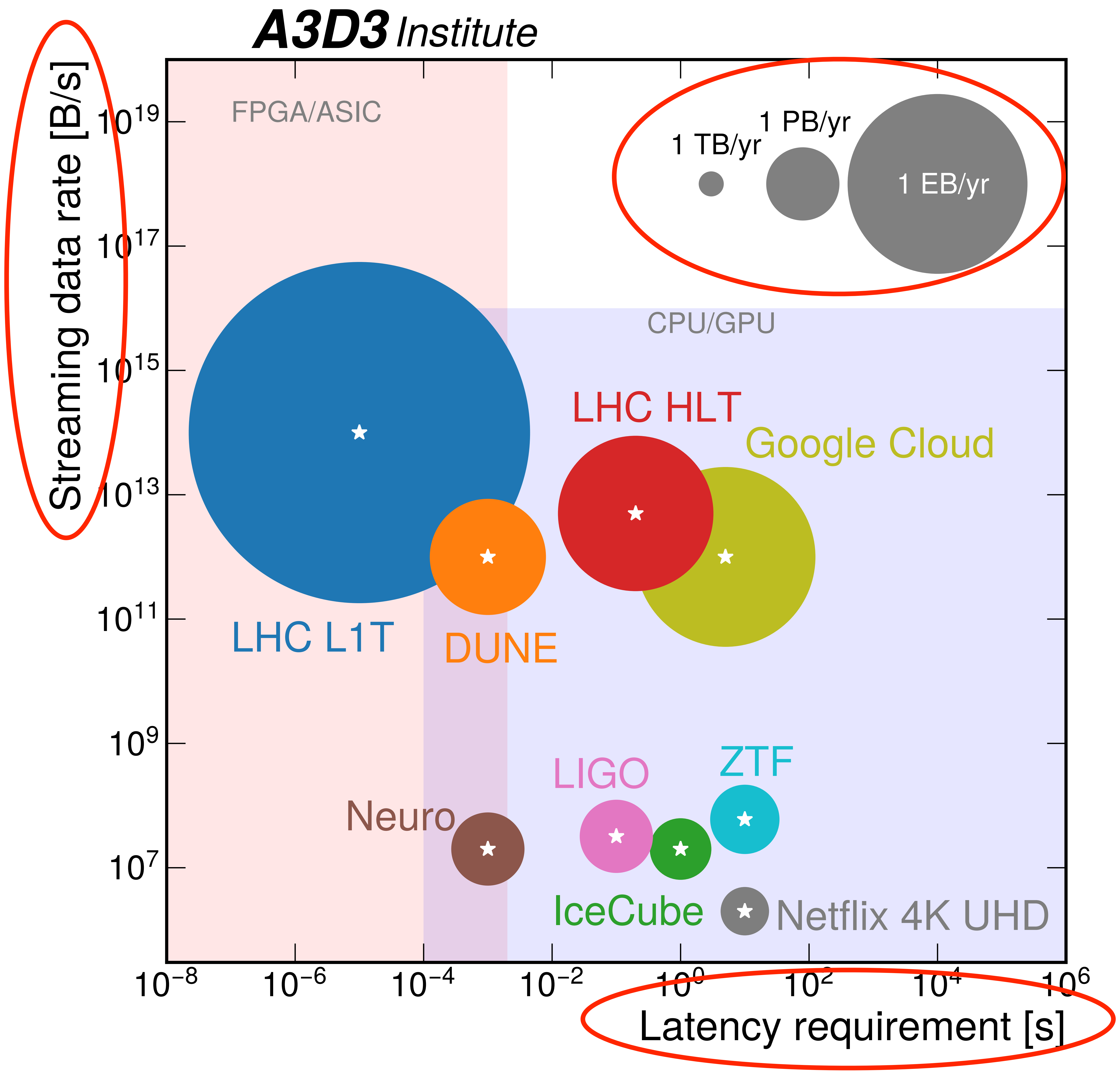




SCIENTIFIC ML CHALLENGES

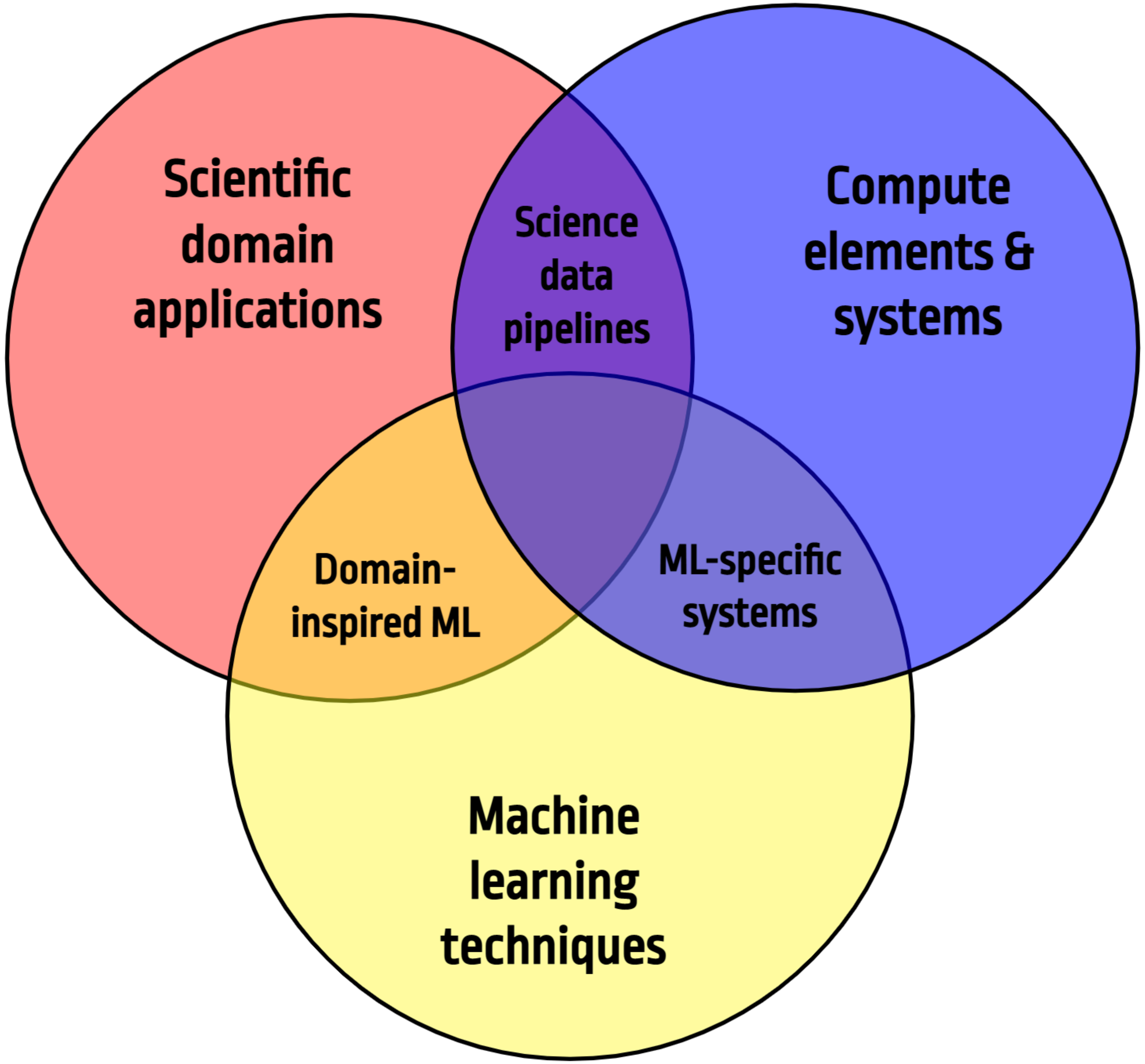


SCIENTIFIC ML CHALLENGES



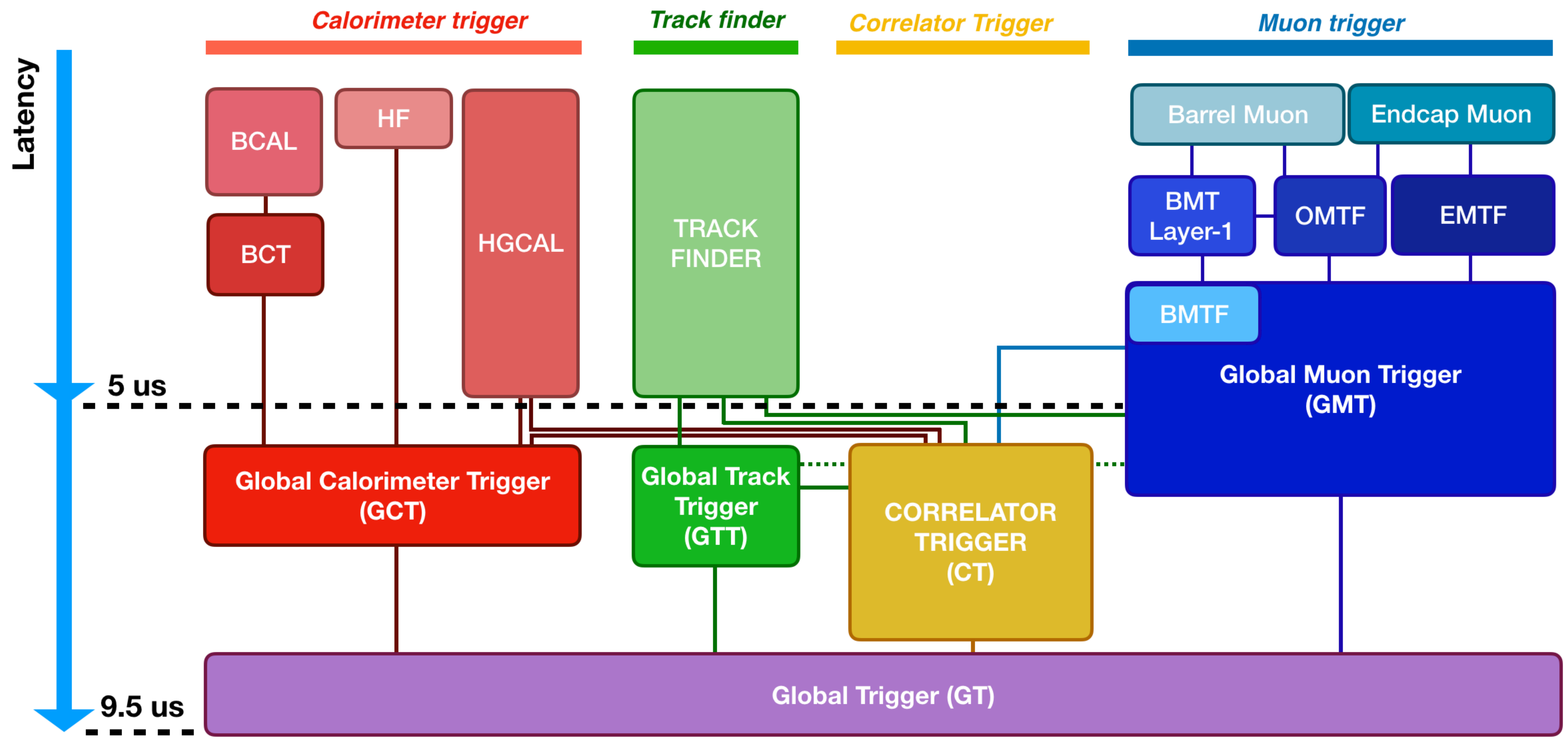
COMMON GOALS

- **Tools**
    - Accessible workflows like HLS to make hardware more accessible domain scientists
- **ML techniques**
    - Efficient training and implementation methods codesigned for specific hardware
- **Hardware**
    - Evolving compute platforms, e.g. power-law growth in FPGA logic



LHC REAL-TIME SYSTEM: LEVEL-1 TRIGGER

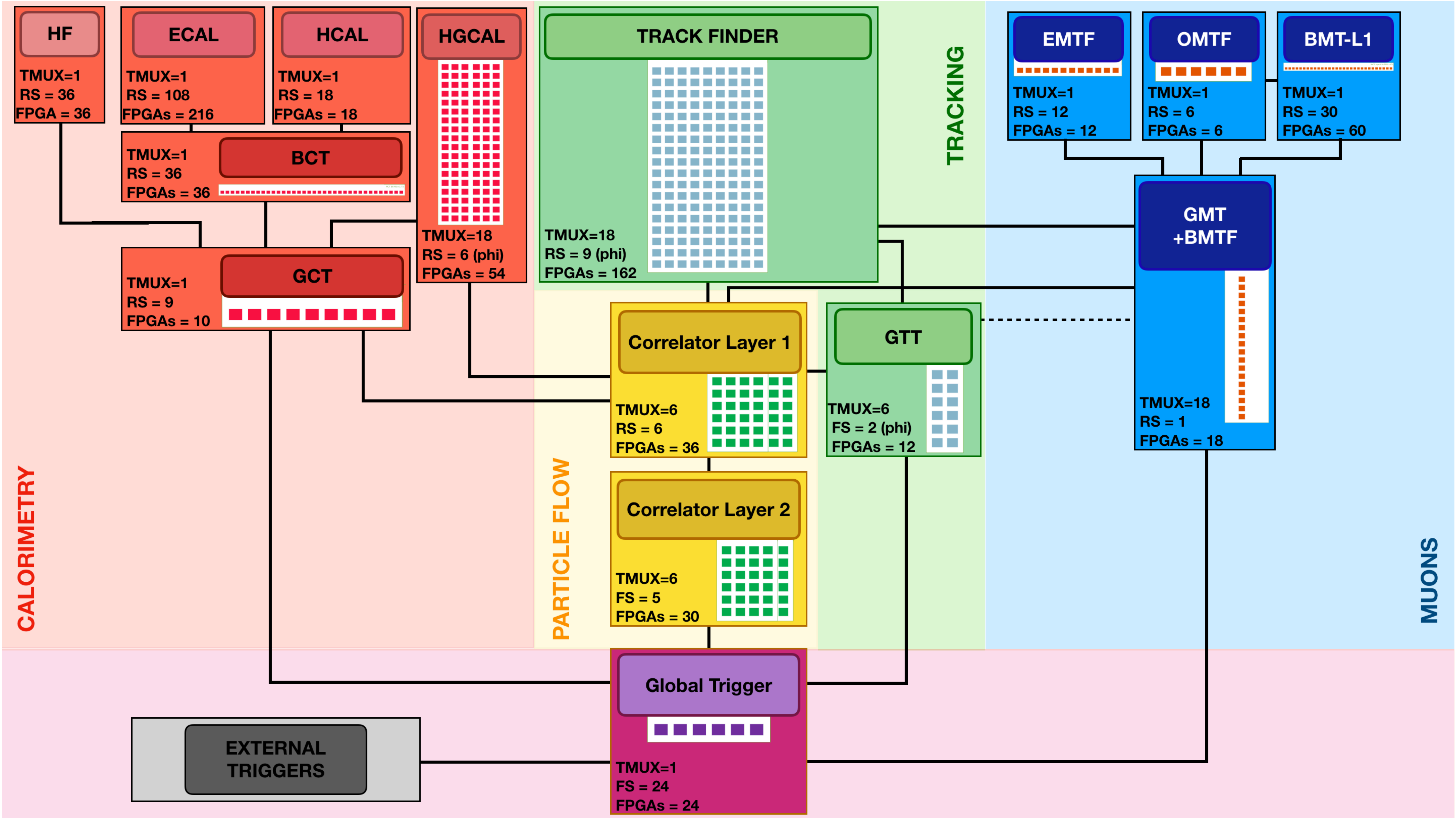
- Reconstruct all events and reject 98% of them in ~12.5 μs
  - Individual algorithms usually have to be < 1 μs and keep up with new events every 25 ns
- Latency necessitates all **FPGA** design (many algorithms running on 729 FPGAs!)
  - Individual algorithms usually have to fit on < 1 FPGA



LHC REAL-TIME SYSTEM: LEVEL-1 TRIGGER

CMS-TDR-021

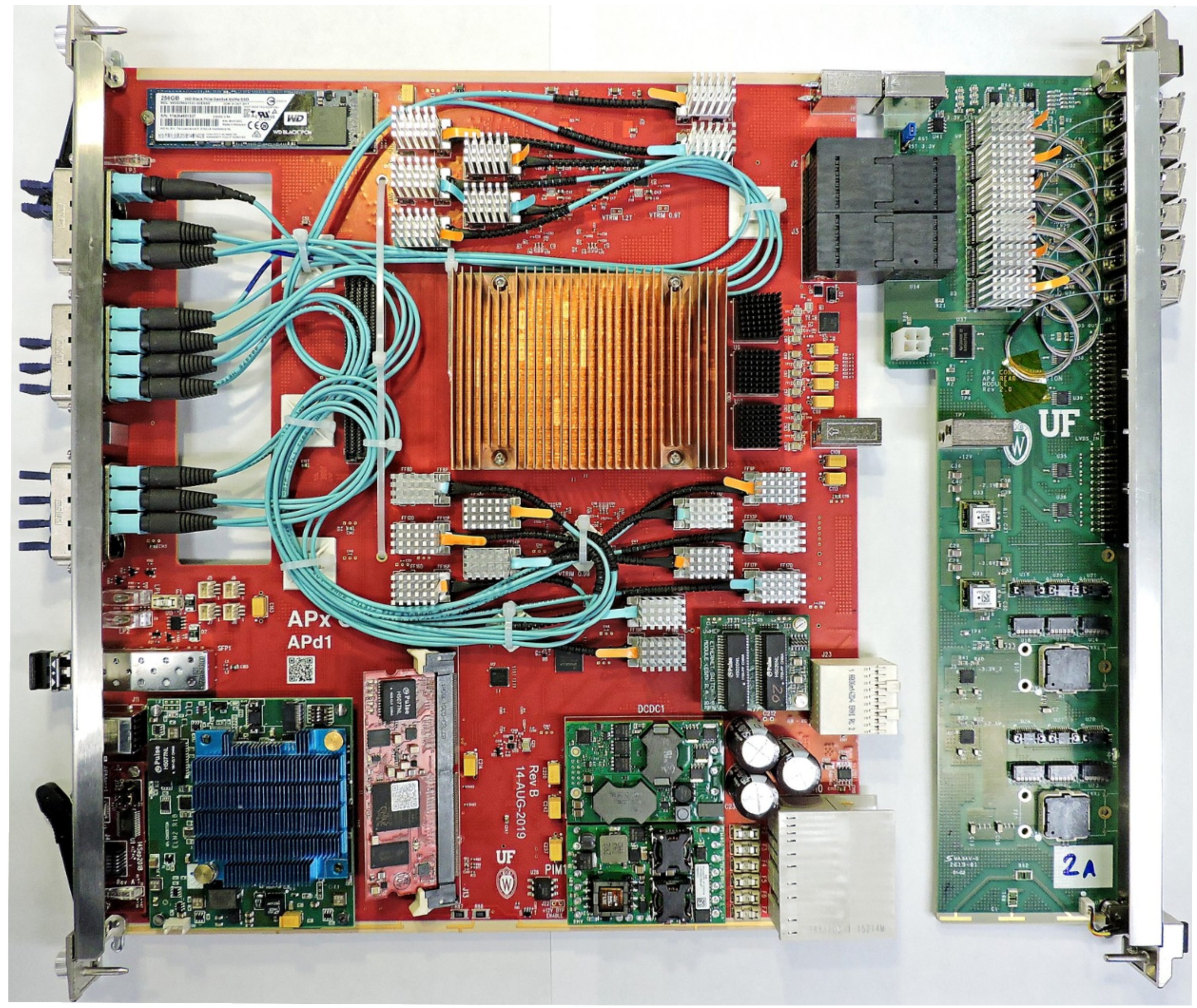
- Reconstruct all events and reject 98% of them in ~12.5 μs
  - Individual algorithms usually have to be < 1 μs and keep up with new events every 25 ns
- Latency necessitates all **FPGA** design (many algorithms running on 729 FPGAs!)
  - Individual algorithms usually have to fit on < 1 FPGA

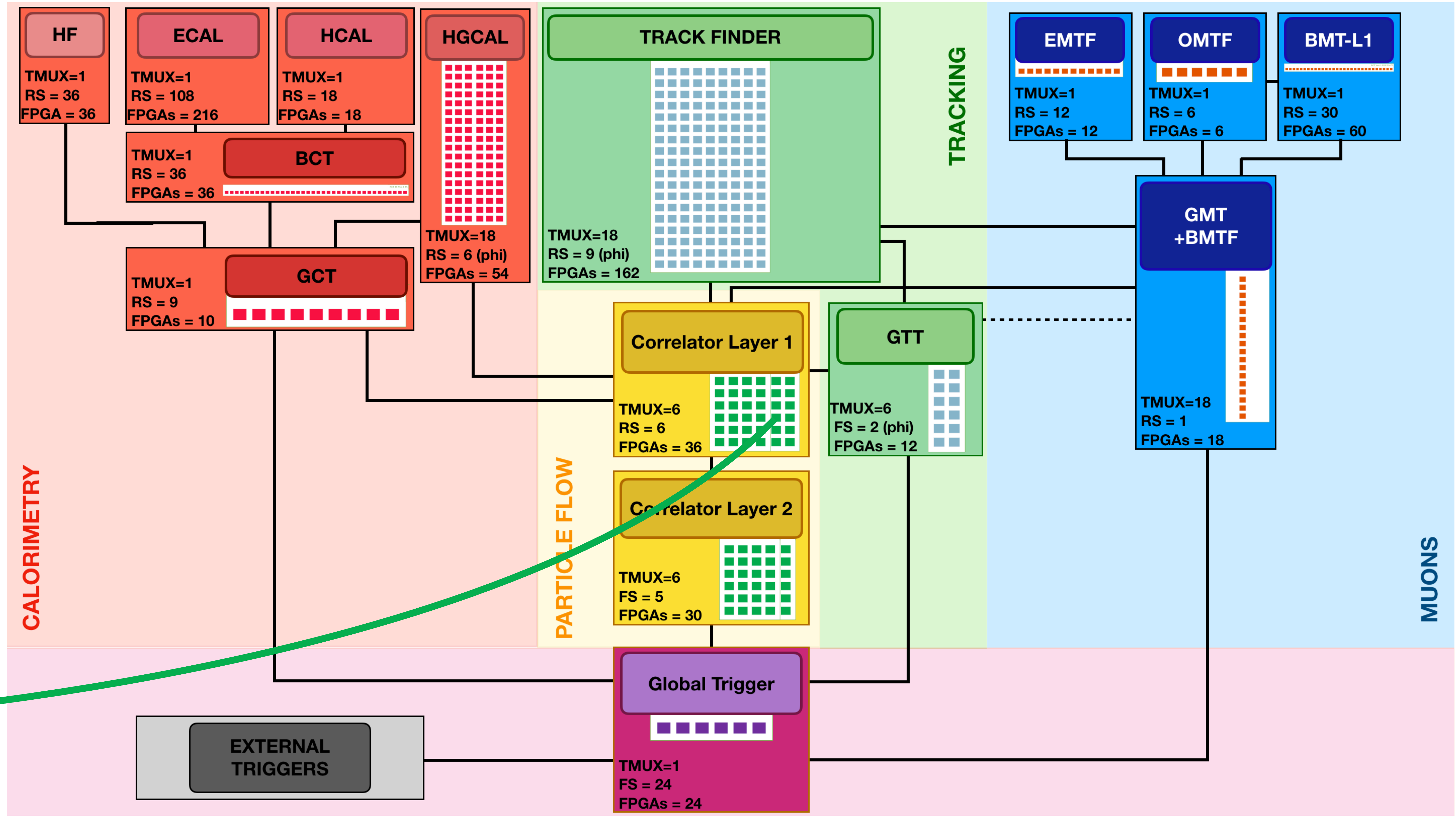


LHC REAL-TIME SYSTEM: LEVEL-1 TRIGGER

CMS-TDR-021

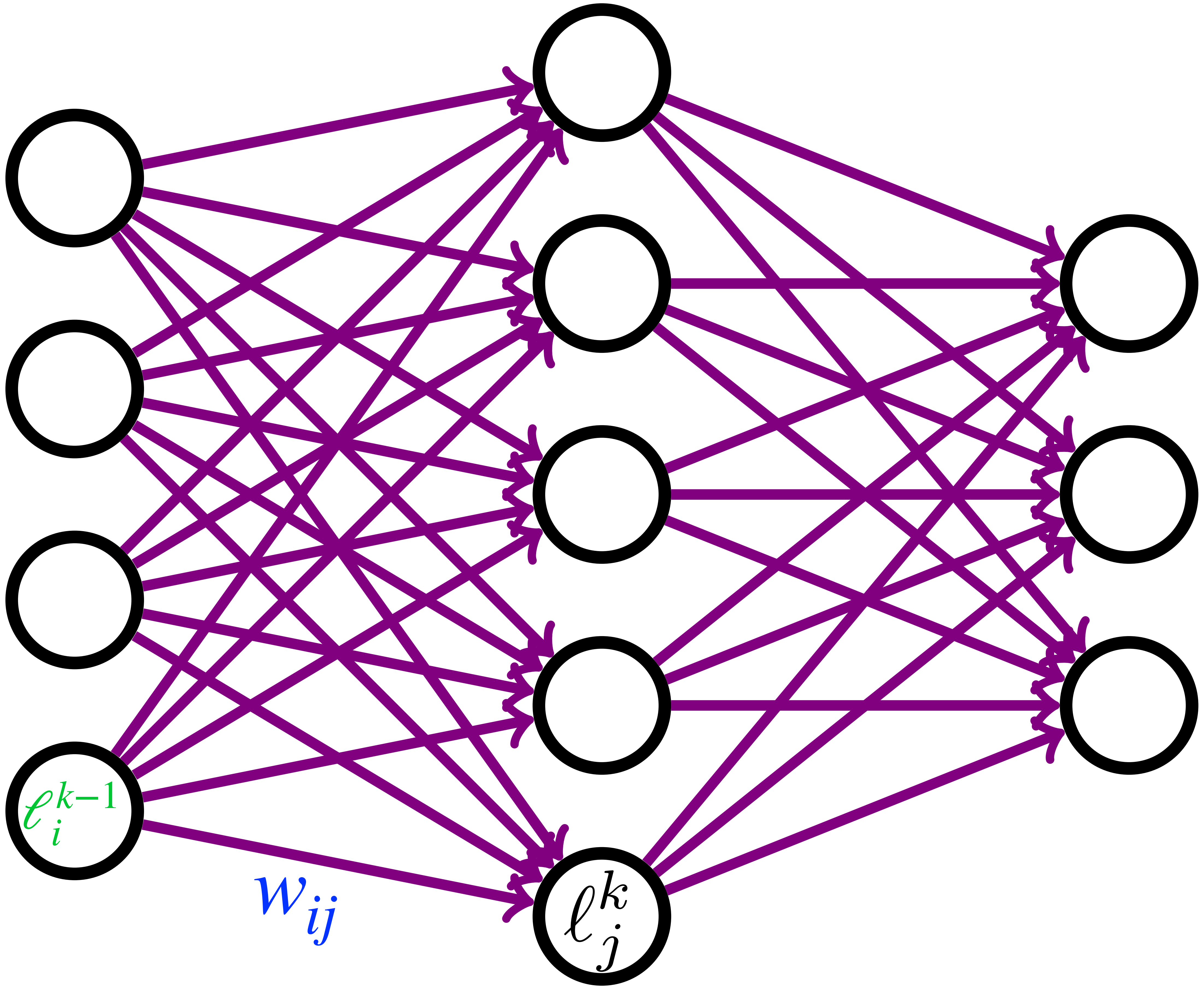
- Reconstruct all events and reject 98% of them in ~12.5 μs
  - Individual algorithms usually have to be < 1 μs and keep up with new events every 25 ns
- Latency necessitates all **FPGA** design (many algorithms running on 729 FPGAs!)
  - Individual algorithms usually have to fit on < 1 FPGA





NEURAL NETWORK OPERATIONS

$$\ell_j^k = \phi\left(\sum_i w_{ij} \ell_i^{k-1} + b_j \right)$$



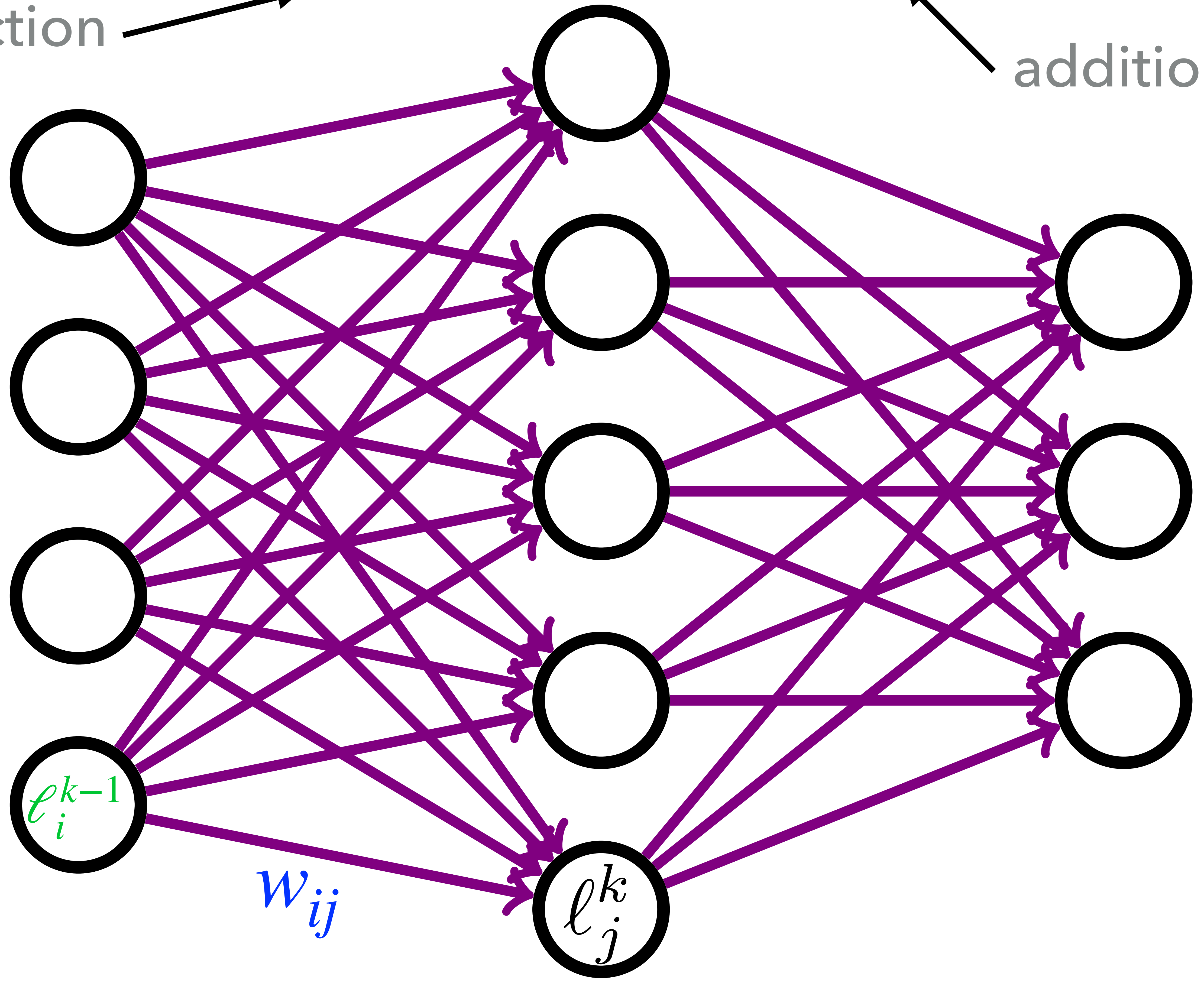
NEURAL NETWORK OPERATIONS

multiplication

$$\ell_j^k = \phi\left(\sum_i w_{ij}\ell_i^{k-1} + b_j\right)$$

activation function

addition



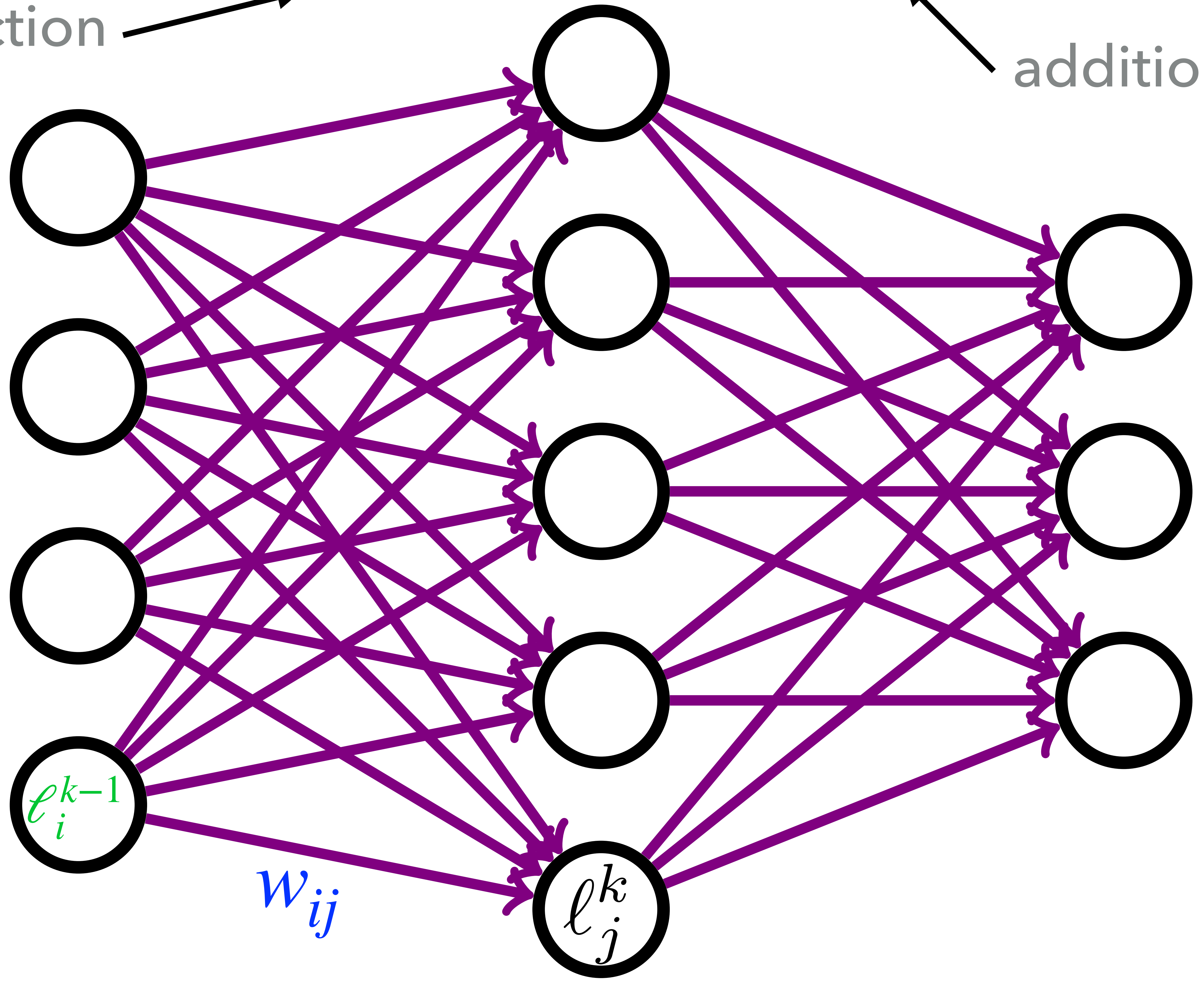
NEURAL NETWORK OPERATIONS

multiplication

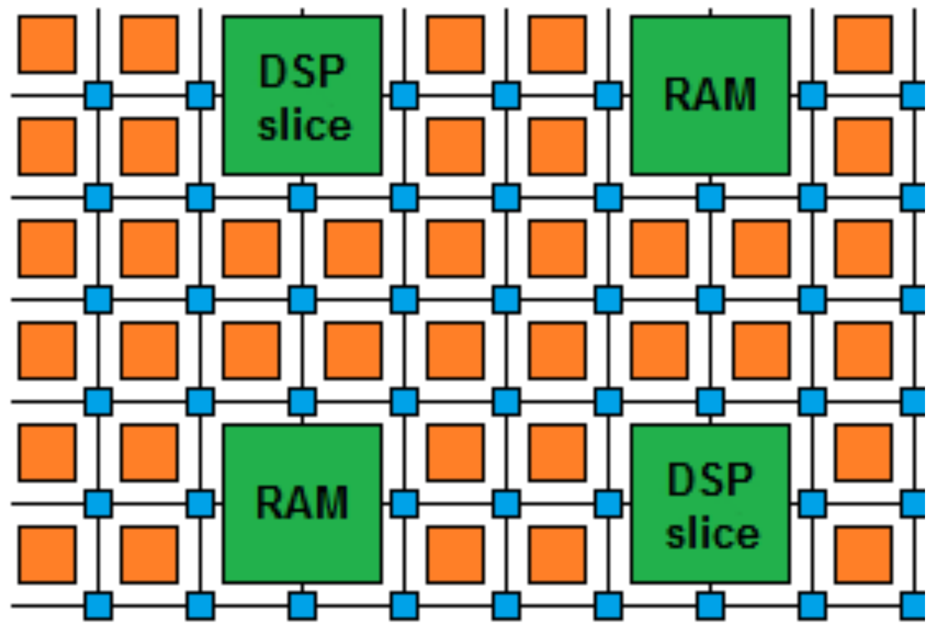
$$\ell_j^k = \phi\left(\sum_i w_{ij}\ell_i^{k-1} + b_j\right)$$

activation function

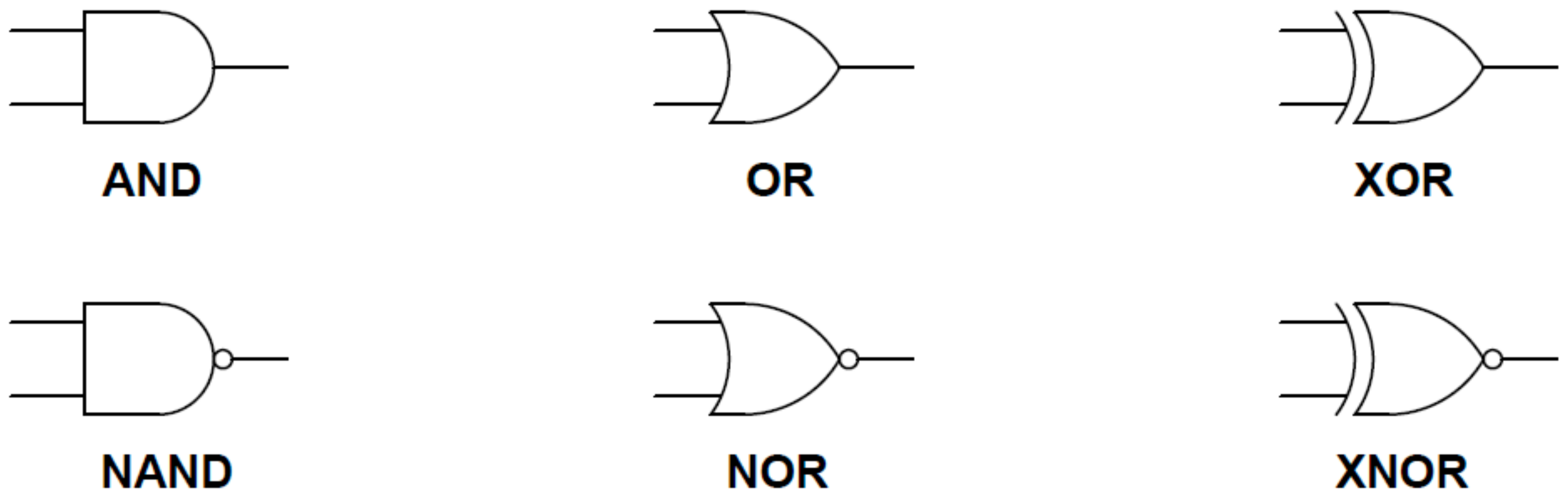
addition



Maps nicely onto FPGA resources: high I/O, DSPs, LUTs, etc.



- Operations can be implemented with core operations (gates)



HOW ARE THEY REPROGRAMMABLE? GATES, LUTS, AND FPGA FABRIC

- Operations can be implemented with core operations (gates)

LUT

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

LUT

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

LUT

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

LUT

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

LUT

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

LUT

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

- Gates are like look-up tables (LUTs)

HOW ARE THEY REPROGRAMMABLE? GATES, LUTS, AND FPGA FABRIC

- Operations can be implemented with core operations (gates)

LUT

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

LUT

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

LUT

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

LUT

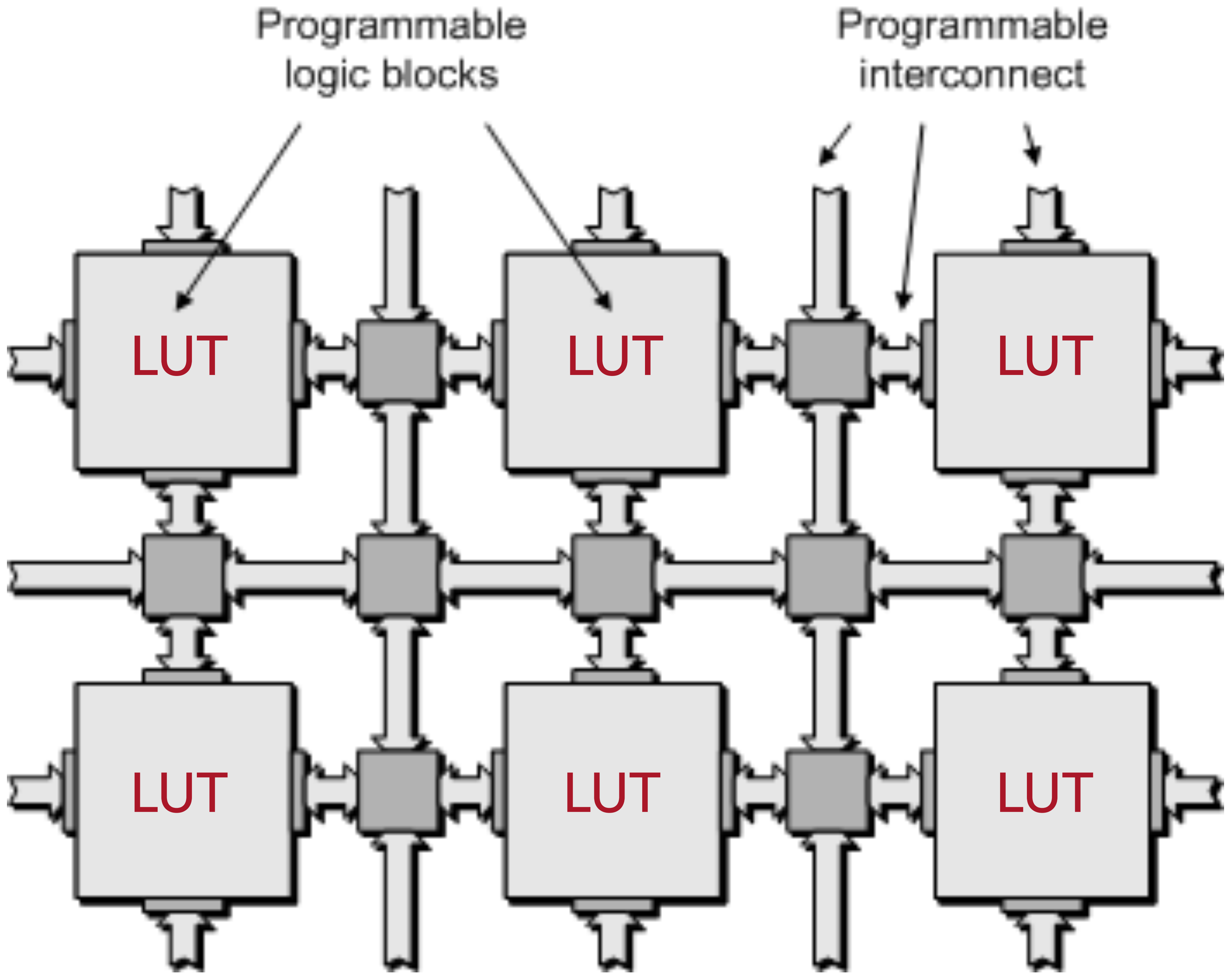
A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

LUT

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

LUT

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

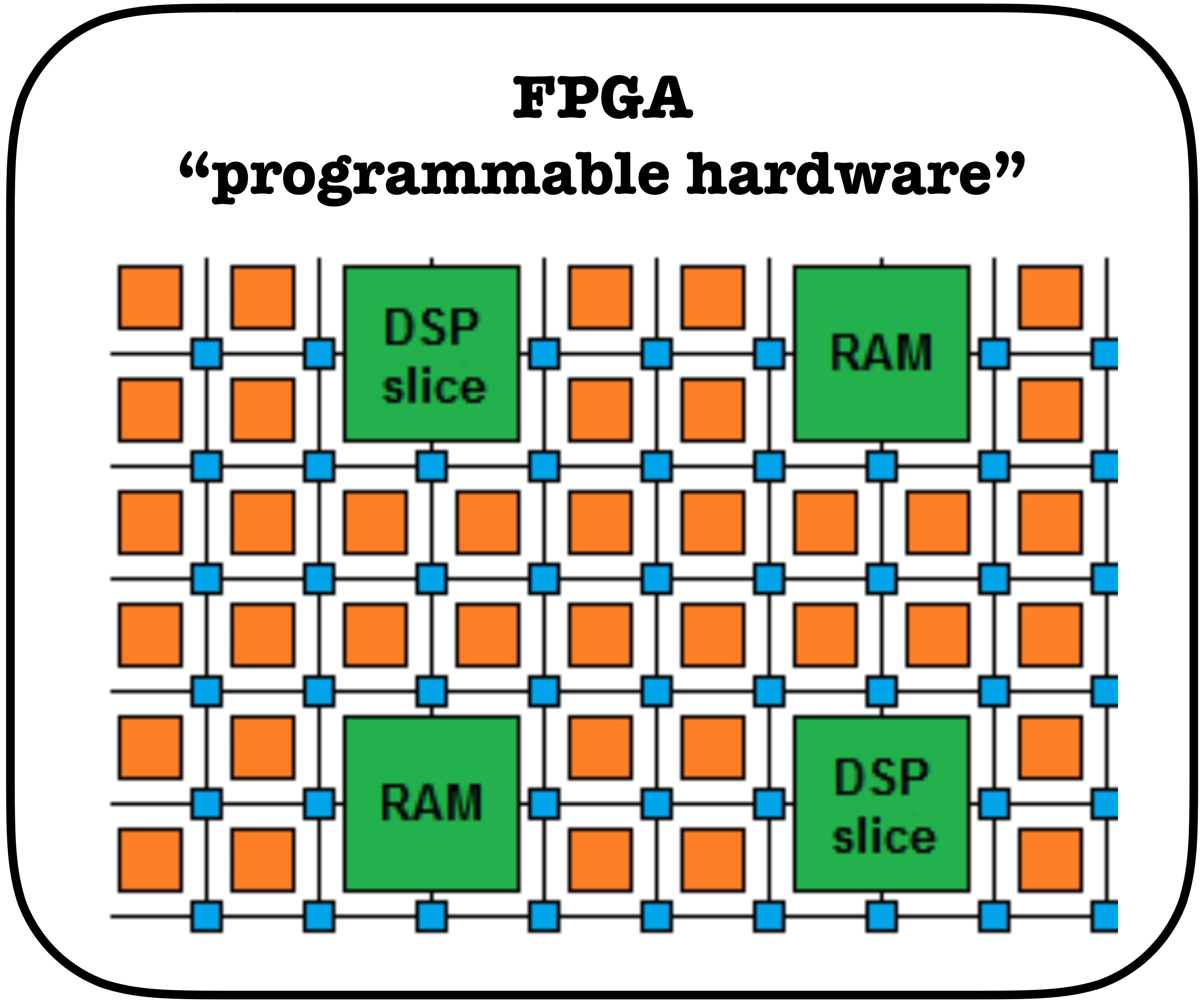


- Gates are like look-up tables (LUTs)

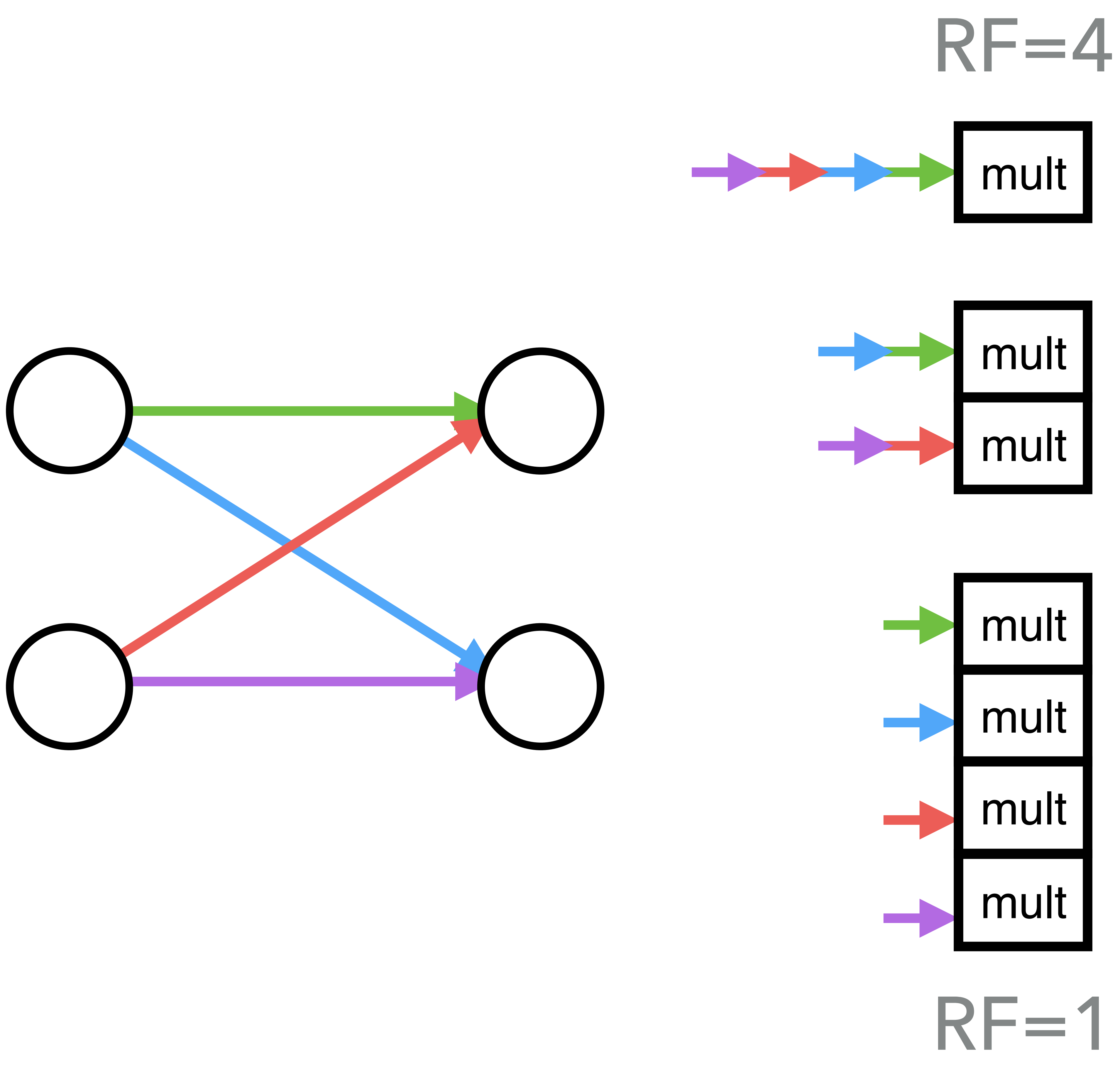
- If we can (re-)program arbitrary LUTs and (re-)connect them however we want, we can (re-)implement whatever algorithm we want!

MODERN FPGAS

- Pros:
  - Reprogrammable interconnects between embedded components that perform multiplication (DSPs), apply logical functions (LUTs), or store memory (BRAM)
  - High throughput I/O: O(100) optical transceivers running at O(15) Gbps
  - Massively parallel
  - Low power
- Cons:
  - Requires domain knowledge to program (using VHDL/Verilog)

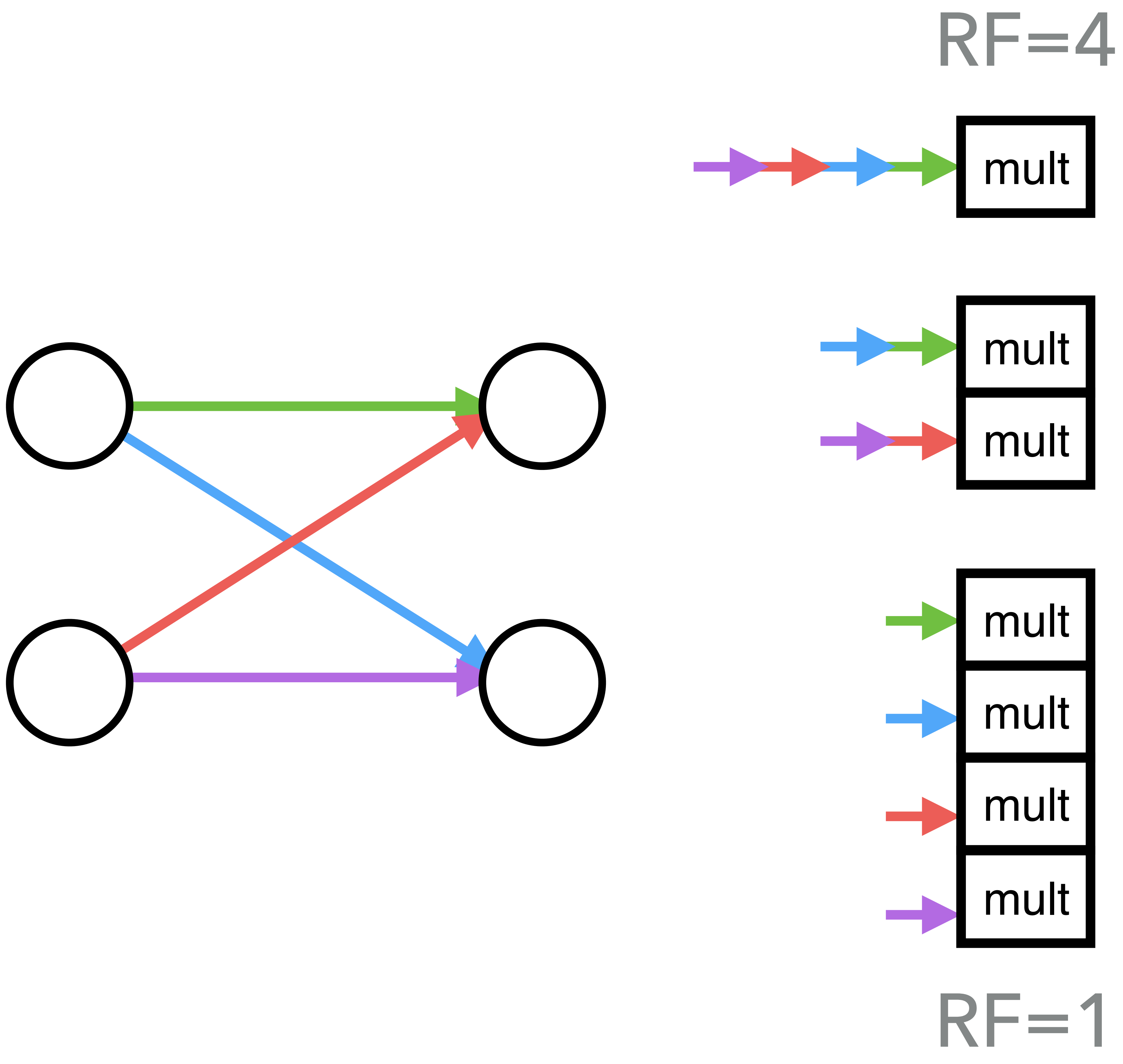


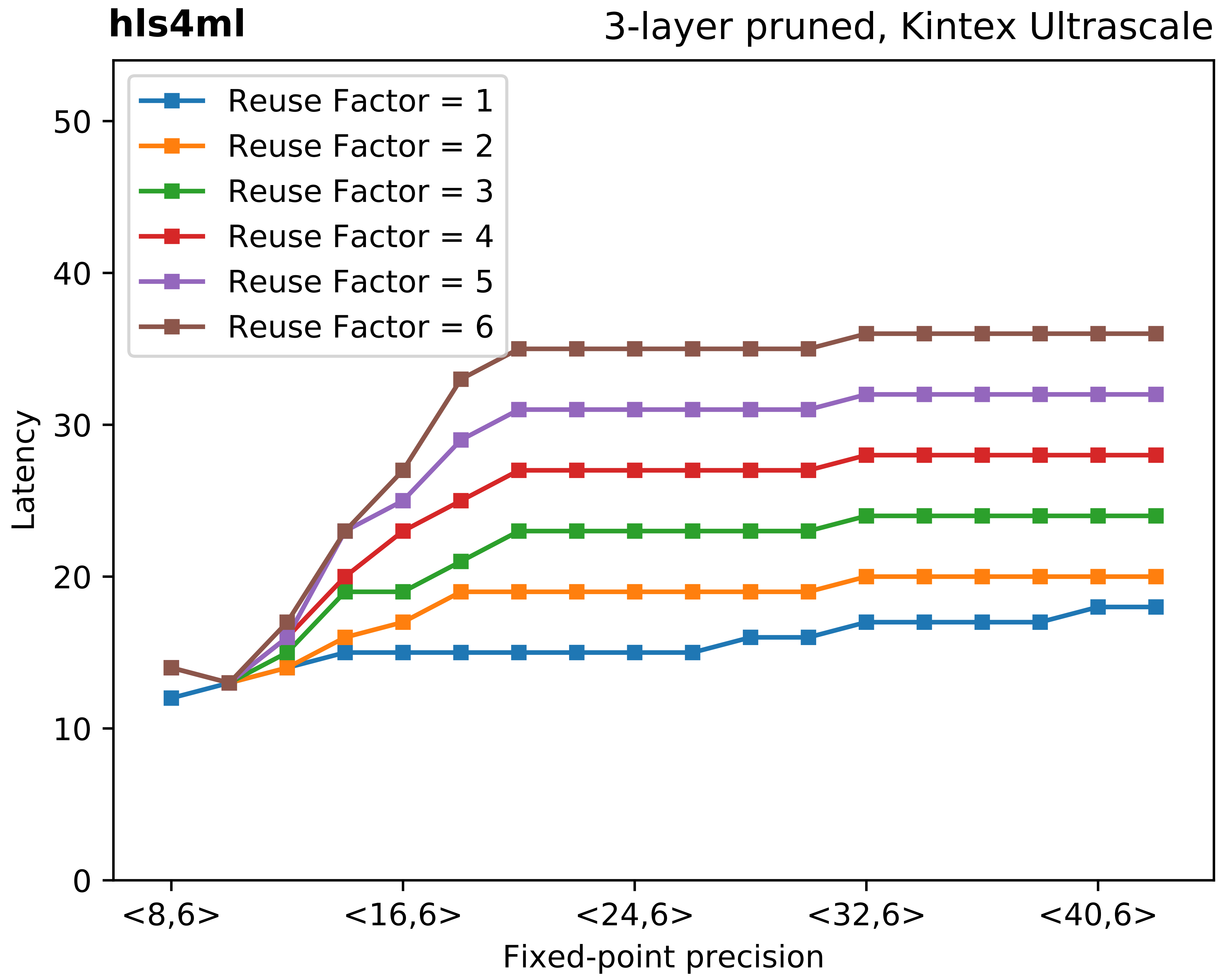
NETWORK TUNING: PARALLELIZATION & TIMING



NETWORK TUNING: PARALLELIZATION & TIMING

- Decreasing reuse factor, increases parallelization and decreases latency





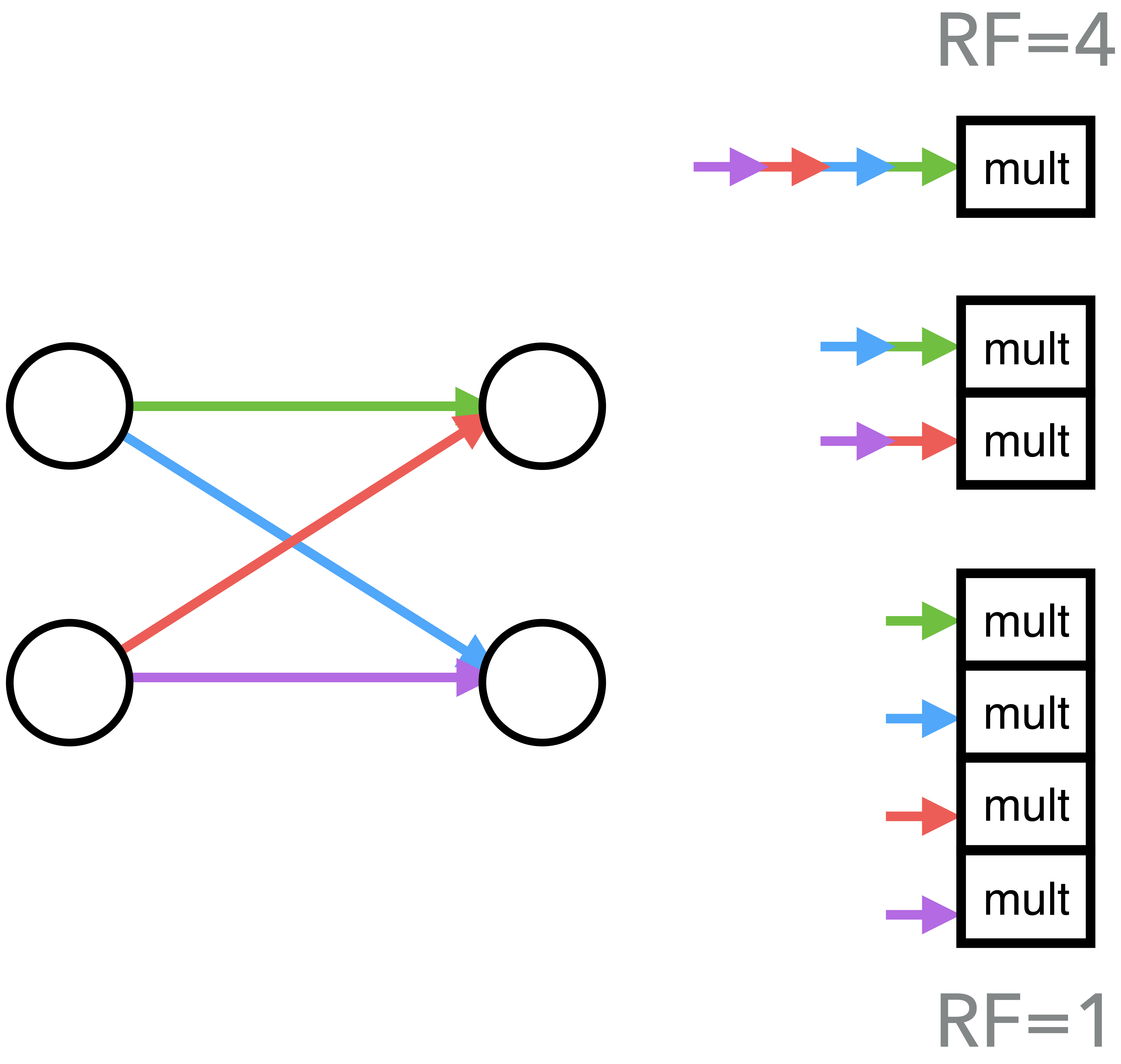
~35 clocks @ 200 MHz = 175 ns

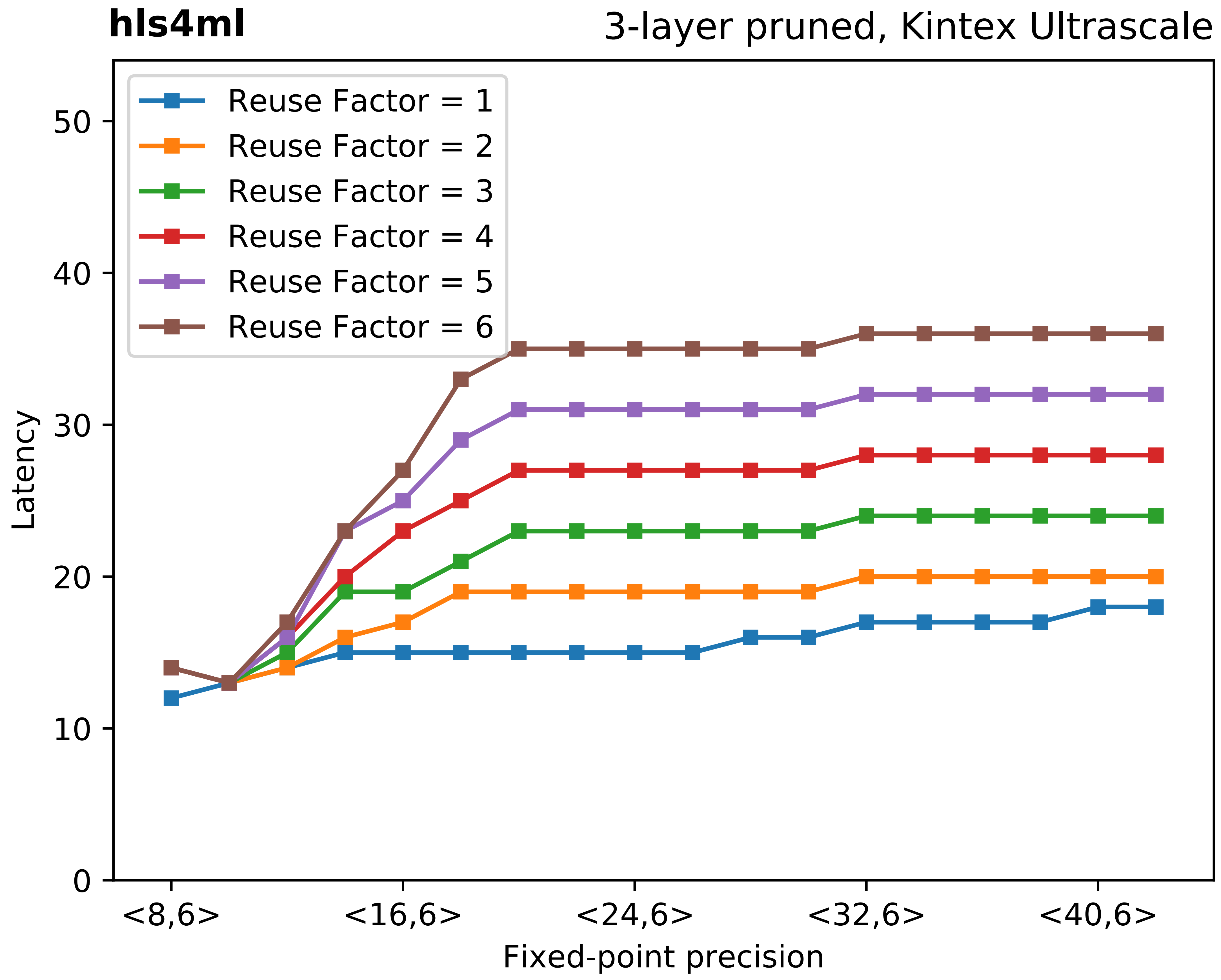
~15 clocks @ 200 MHz = 75 ns

NETWORK TUNING: PARALLELIZATION & TIMING

J. Instrum. 13, P07027 (2018)

- Decreasing reuse factor, increases parallelization and decreases latency





~35 clocks @ 200 MHz = 175 ns

~15 clocks @ 200 MHz = 75 ns

- Algorithm comfortably fits in latency requirements (<1 μs)

APPLICATIONS OF FAST ML

Too many results to cover!

ULTRA-LOW LATENCY RECURRENT NEURAL NETWORK INFERENCE ON FPGAS FOR PHYSICS APPLICATIONS WITH HLS4ML

JINST
Published by IOP Publishing for Sissa Medialab
Received: May 10, 2018
Accepted: July 17, 2018
Published: July 27, 2018

Fast inference of deep neural networks in FPGAs for particle physics

J. Duarte,[a] S. Han,[b] P. Harris,[b] S. Jindariani,[a] E. Kreinar,[c] B. Kreis,[a] J. Ngadiuba,[d] M. Pierini,[d] R. Rivera,[a] N. Tran[a,1] and Z. Wu[e]

frontiers in Big Data
ORIGINAL RESEARCH
published: 12 January 2021
doi: 10.3389/fdata.2020.598927

Distance-Weighted Graph Neural Networks on FPGAs for Real-Time Particle Reconstruction in High Energy Physics

JINST
Published by IOP Publishing for Sissa Medialab
Received: February 20, 2020
Accepted: April 7, 2020
Published: May 29, 2020

Fast inference of Boosted Decision Trees in FPGAs for particle physics

S. Summers,[a,1] G. Di Guglielmo,[b] J. Duarte,[c] P. Harris,[d] D. Hoang,[e] S. Jindariani,[f] E. Kreinar,[g] V. Loncar,[a,h] J. Ngadiuba,[a] M. Pierini,[a] D. Rankin,[d] N. Tran[f] and Z. Wu[i]

ARTICLES
https://doi.org/10.1038/s42256-022-00441-3
nature machine intelligence
Check for updates

Autoencoders on field-programmable gate arrays for real-time, unsupervised new physics detection at 40 MHz at the Large Hadron Collider

Ekaterina Govorkova[1]✉, Ema Puljak[1], Thea Aarrestad[1], Thomas James[1], Vladimir Loncar[1,2], Maurizio Pierini[1], Adrian Alan Pol[1], Nicolò Ghielmetti[1,3], Maksymilian Graczyk[1,4], Sioni Summers[1], Jennifer Ngadiuba[5,6], Thong Q. Nguyen[6], Javier Duarte[7] and Zhenbin Wu[8]

arXiv:2103.05579v3 [cs.LG] 23 Mar 2021

hls4ml: An Open-Source Codesign Workflow to Empower Scientific Low-Power Machine Learning Devices

Farah Fahim*, Benjamin Hawks, Christian Herwig, James Hirschauer, Sergo Jindariani, Nhan Tran* — Fermilab, Batavia, IL, USA
Luca P. Carloni, Giuseppe Di Guglielmo — Columbia University, New York, NY, USA
Philip Harris, Jeffrey Krupa, Dylan Rankin — MIT, Cambridge, MA, USA
Manuel Blanco Valentin, Josiah Hester, Yingyi Luo, John Mamish, Seda Orgrenci-Memik — Northwestern University, Evanston, IL, USA
Thea Aarrestad, Hamza Javed, Vladimir Loncar, Maurizio Pierini, Adrian Alan Pol, Sioni Summers — European Organization for Nuclear Research (CERN), Geneva, Switzerland
Javier Duarte — UC San Diego, La Jolla, CA, USA, jduarte@ucsd.edu
Scott Hauck, Shih-Chieh Hsu — University of Washington, Seattle, WA, USA
Jennifer Ngadiuba — Caltech, Pasadena, CA, USA
Mia Liu — Purdue University, West Lafayette, IN, USA
Duc Hoang — Rhodes College, Memphis, TN, USA
Edward Kreinar — HawkEye360, Herndon, VA, USA
Zhenbin Wu — University of Illinois at Chicago, Chicago, IL, USA

nature machine intelligence
ARTICLES
https://doi.org/10.1038/s42256-021-00356-5
Check for updates

Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors

Claudionor N. Coelho Jr[1], Aki Kuusela[2], Shan Li[2], Hao Zhuang[2], Jennifer Ngadiuba[3], Thea Klaeboe Aarrestad[4]✉, Vladimir Loncar[4,5], Maurizio Pierini[4], Adrian Alan Pol[4] and Sioni Summers[4]

MACHINE LEARNING Science and Technology

Compressing deep neural networks on FPGAs to binary and ternary precision with `hls4ml`

Jennifer Ngadiuba[1], Vladimir Loncar[1], Maurizio Pierini[1], Sioni Summers[1], Giuseppe Di Guglielmo[2], Javier Duarte[3], Philip Harris[4], Dylan Rankin[4], Sergo Jindariani[5], Mia Liu[5], Kevin Pedro[5], Nhan Tran[5], Edward Kreinar[6], Sheila Sagear[7], Zhenbin Wu[8] and Duc Hoang[9]

CERN European Organization for Nuclear Research
Organisation européenne pour la recherche nucléaire
CERN-LHCC-2020-004
CMS-TDR-021
10 March 2020

The Phase-2 Upgrade of the CMS Level-1 Trigger Technical Design Report

A Reconfigurable Neural Network ASIC for Detector Front-End Data Compression at the HL-LHC

Giuseppe Di Guglielmo, Farah Fahim, *Member, IEEE*, Christian Herwig, Manuel Blanco Valentin, Javier Duarte, Cristian Gingu, *Member, IEEE*, Philip Harris, James Hirschauer, Martin Kwok, Vladimir Loncar, Yingyi Luo, Llovizna Miranda, Jennifer Ngadiuba, Daniel Noonan, Seda Ogrenci-Memik, Maurizio Pierini, Sioni Summers, and Nhan Tran

ESP4ML: Platform-Based Design of Systems-on-Chip for Embedded Machine Learning

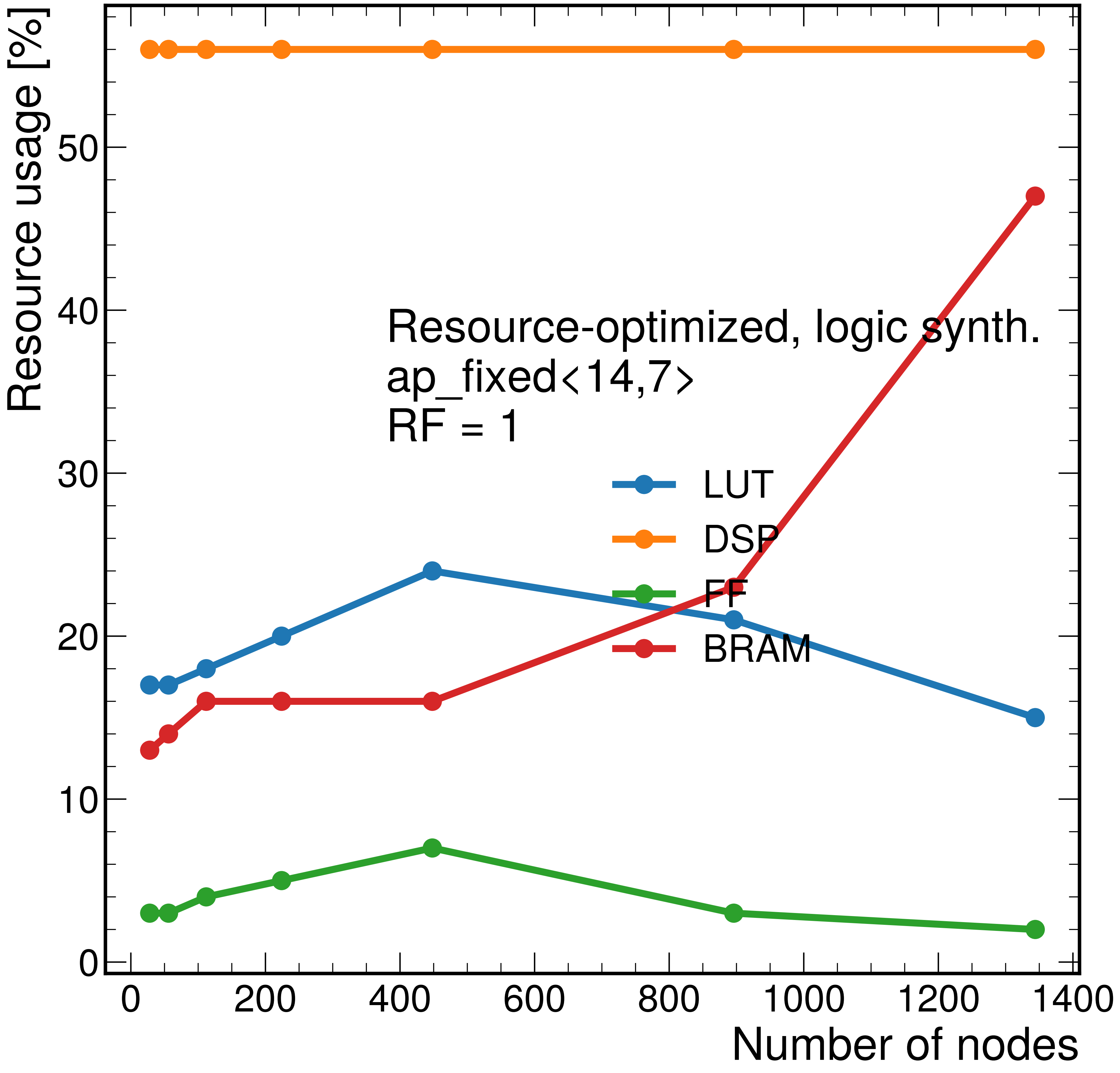
Davide Giri, Kuan-Lin Chiu, Giuseppe Di Guglielmo, Paolo Mantovani and Luca P. Carloni
Department of Computer Science · Columbia University, New York
[davide_giri, chiu, giuseppe, paolo, luca]@cs.columbia.edu

frontiers in Big Data
ORIGINAL RESEARCH
published: 23 March 2022
doi: 10.3389/fdata.2022.828666

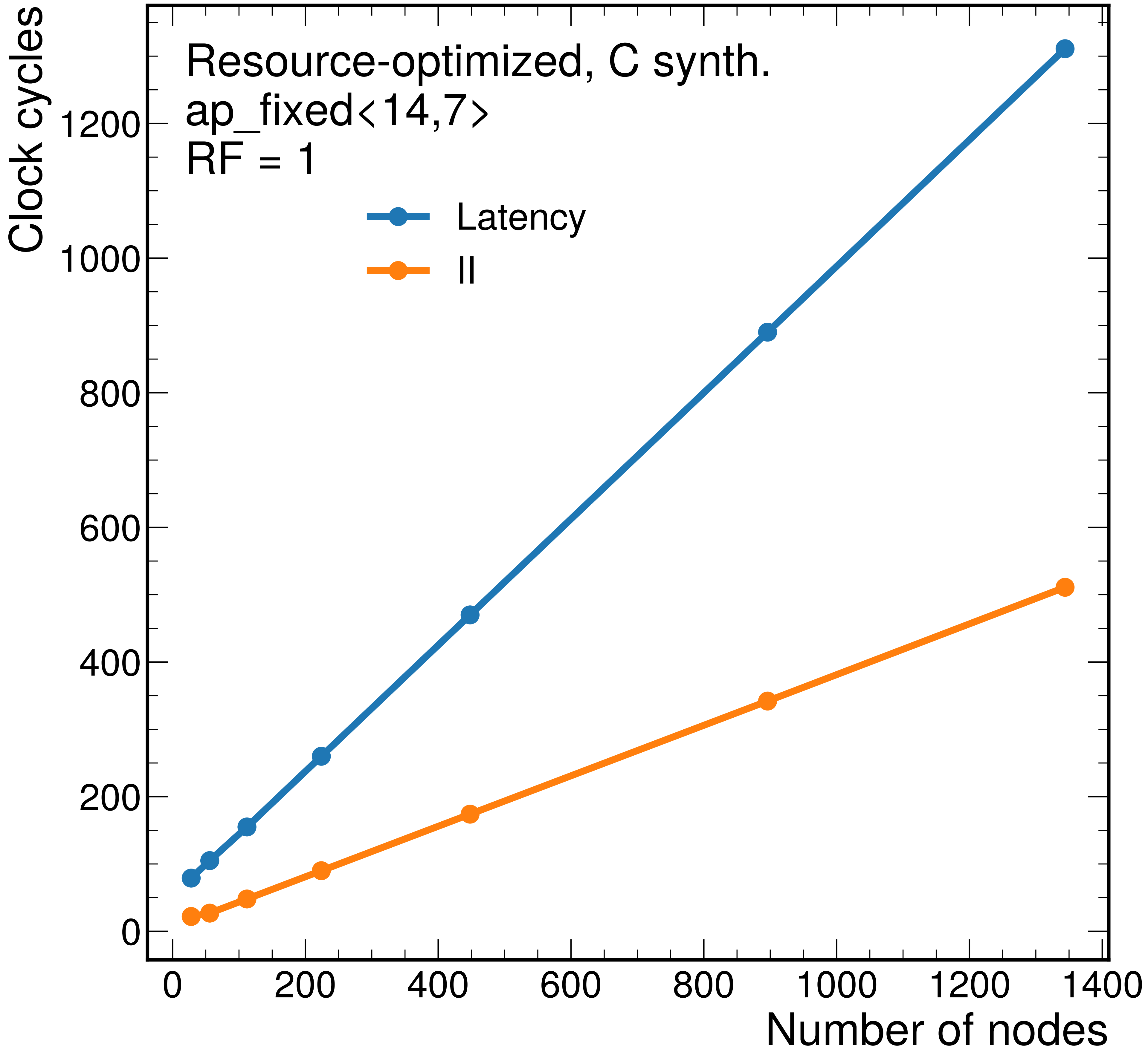
Graph Neural Networks for Charged Particle Tracking on FPGAs

Abdelrahman Elabd[1], Vesal Razavimaleki[2], Shi-Yu Huang[3], Javier Duarte[2], Markus Atkinson[4], Gage DeZoort[5], Peter Elmer[5], Scott Hauck[6], Jin-Xuan Hu[3], Shih-Chieh Hsu[6,7], Bo-Cheng Lai[3], Mark Neubauer[4*], Isobel Ojalvo[5], Savannah Thais[5] and Matthew Trahms[6]*

SCALING TO LARGER GRAPHS

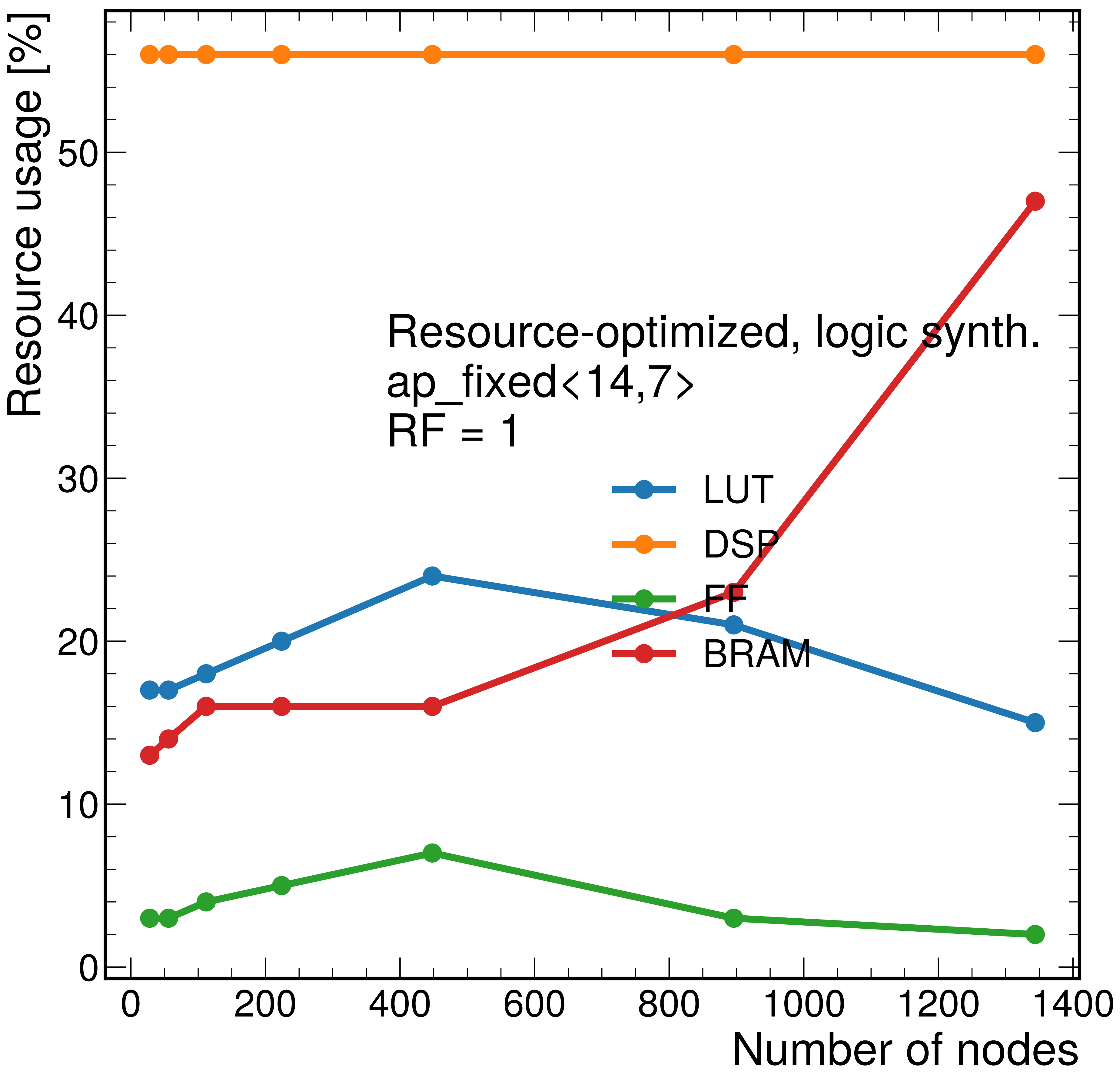


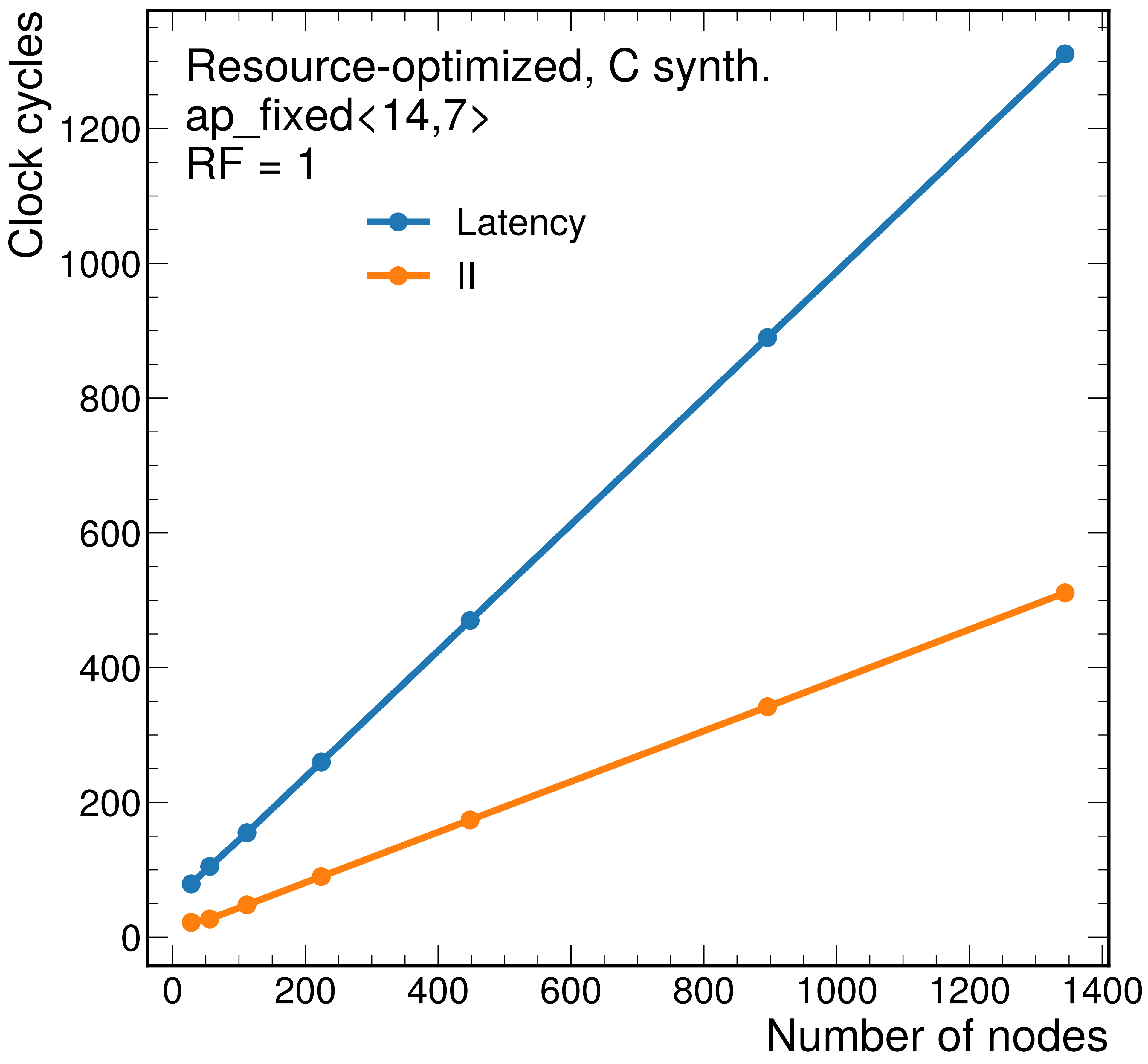
1 clock cycle = 5 ns



SCALING TO LARGER GRAPHS

1 clock cycle = 5 ns





- Modified design can scale to much larger graphs (~1400 nodes, ~2800 edges), for longer latency (6 μs) and II (2 μs)

FAST ML SCIENCE BENCHMARKS

1. Define generic ML benchmarks for bespoke domain problems that attract interest from a broad community of system and ML experts
2. Design benchmarks to satisfy challenging scientific requirements that overlap with a number of systems

FAST ML SCIENCE BENCHMARKS

1. Define generic ML benchmarks for bespoke domain problems that attract interest from a broad community of system and ML experts
2. Design benchmarks to satisfy challenging scientific requirements that overlap with a number of systems

FASTML SCIENCE BENCHMARKS:
ACCELERATING REAL-TIME SCIENTIFIC EDGE MACHINE LEARNING

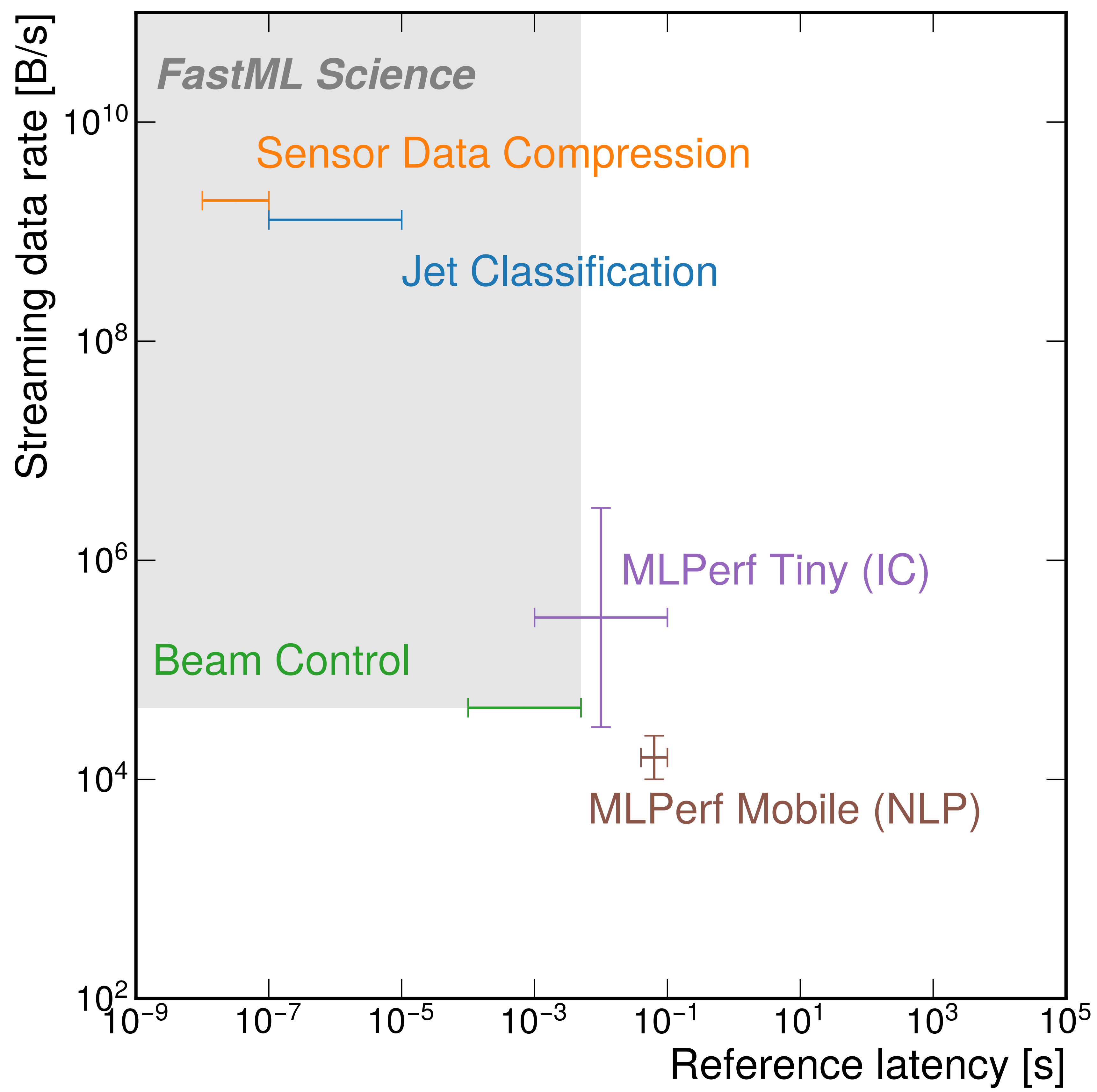
Javier Duarte [*1] Nhan Tran [*2] Ben Hawks [2] Christian Herwig [2]
Jules Muhizi [3] Shvetank Prakash [3] Vijay Janapa Reddi [3]

FAST ML SCIENCE BENCHMARKS

1. Define generic ML benchmarks for bespoke domain problems that attract interest from a broad community of system and ML experts
2. Design benchmarks to satisfy challenging scientific requirements that overlap with a number of systems
   - Set of 3 benchmarks inspired by low-latency edge ML use cases in science
   - Cover a wide range of latency/data rate constraints

FASTML SCIENCE BENCHMARKS: ACCELERATING REAL-TIME SCIENTIFIC EDGE MACHINE LEARNING

Javier Duarte[*1] Nhan Tran[*2] Ben Hawks[2] Christian Herwig[2]
Jules Muhizi[3] Shvetank Prakash[3] Vijay Janapa Reddi[3]



FAST ML SCIENCE BENCHMARKS

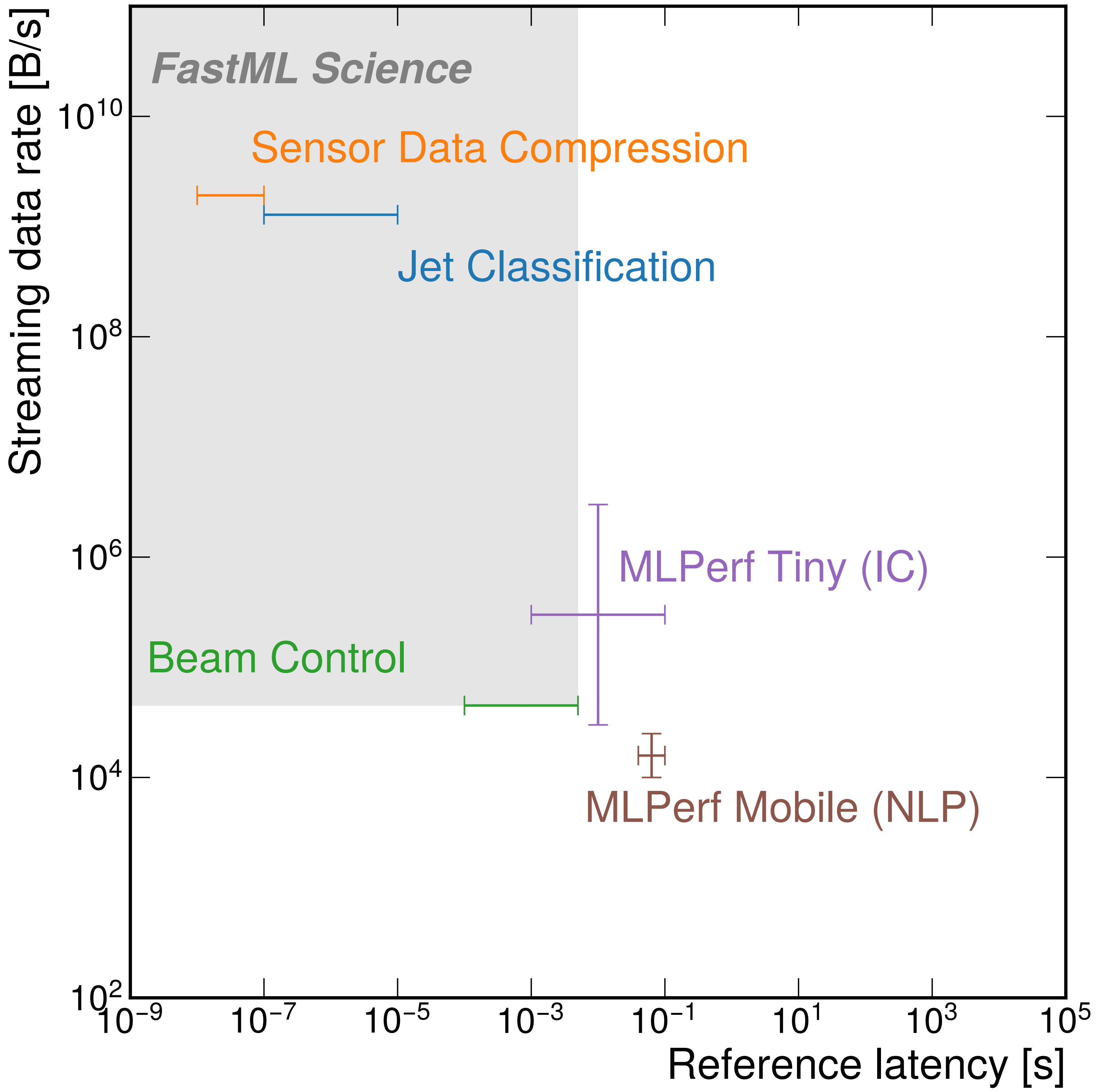
1. Define generic ML benchmarks for bespoke domain problems that attract interest from a broad community of system and ML experts
2. Design benchmarks to satisfy challenging scientific requirements that overlap with a number of systems

- Set of 3 benchmarks inspired by low-latency edge ML use cases in science
- Cover a wide range of latency/data rate constraints
- Unique set of qualities

FASTML SCIENCE BENCHMARKS: ACCELERATING REAL-TIME SCIENTIFIC EDGE MACHINE LEARNING

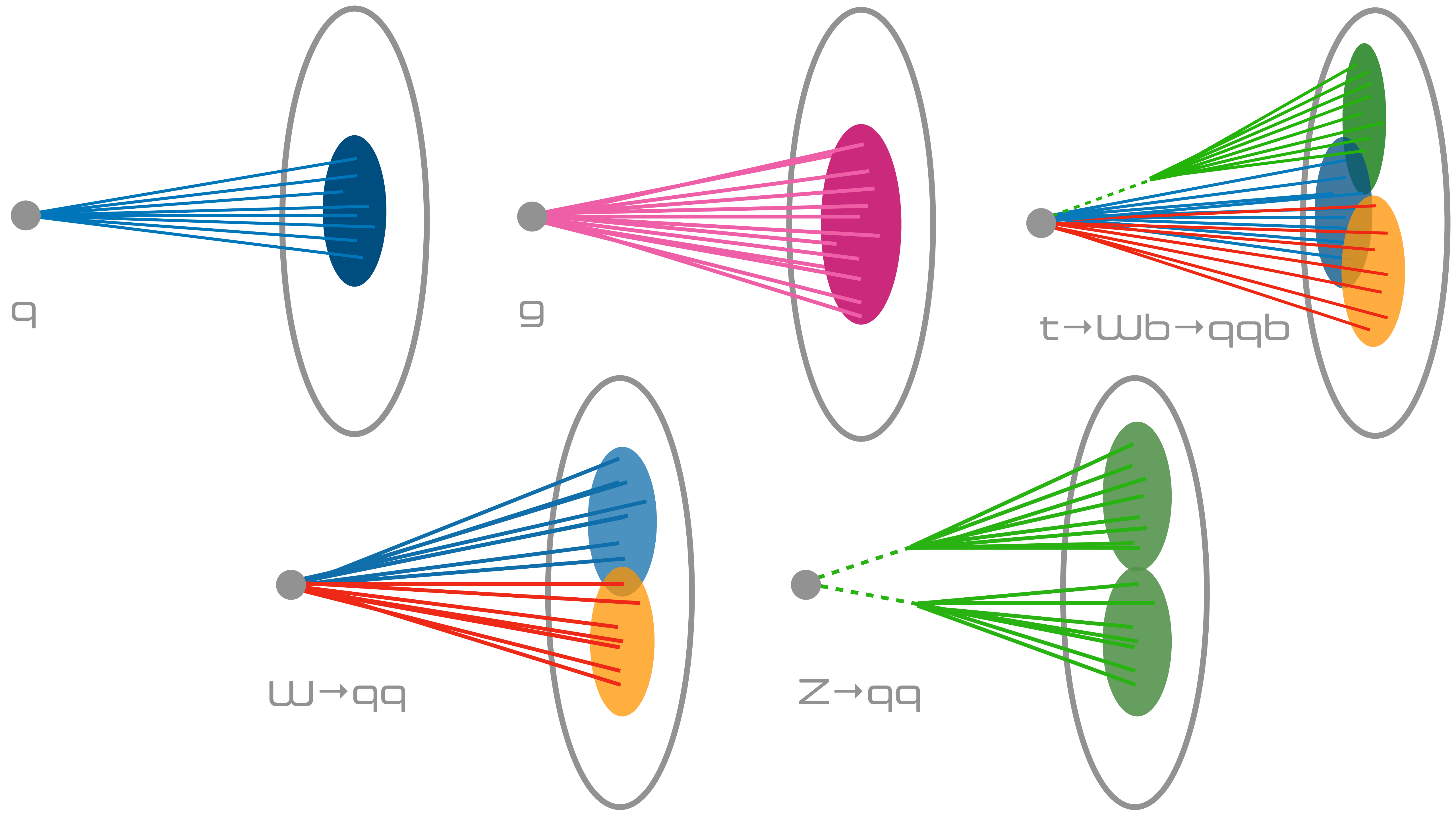
Javier Duarte [*1] Nhan Tran [*2] Ben Hawks [2] Christian Herwig [2]
Jules Muhizi [3] Shvetank Prakash [3] Vijay Janapa Reddi [3]

	Formalized Benchmark	Scientific Workload(s)	Edge Computing	Real-Time Constraints
FastML Science Benchmarks (this work)	✓	✓	✓	✓
SciMLBench (Thiyagalingam et al., 2021)	✓	✓	✓	x
LHC New Physics Dataset (Govorkova et al., 2021)	x	✓	✓	✓
MLPerf HPC (Farrell et al., 2021)	✓	✓	x	x
BenchCounil AIBench HPC (BenchCouncil, 2018)	✓	✓	x	x
MLCommons Science (MLCommons, 2020)	✓	✓	x	x
ITU Modulation Classification (ITU, 2021)	x	x	✓	✓



CURRENT & FUTURE BENCHMARKS

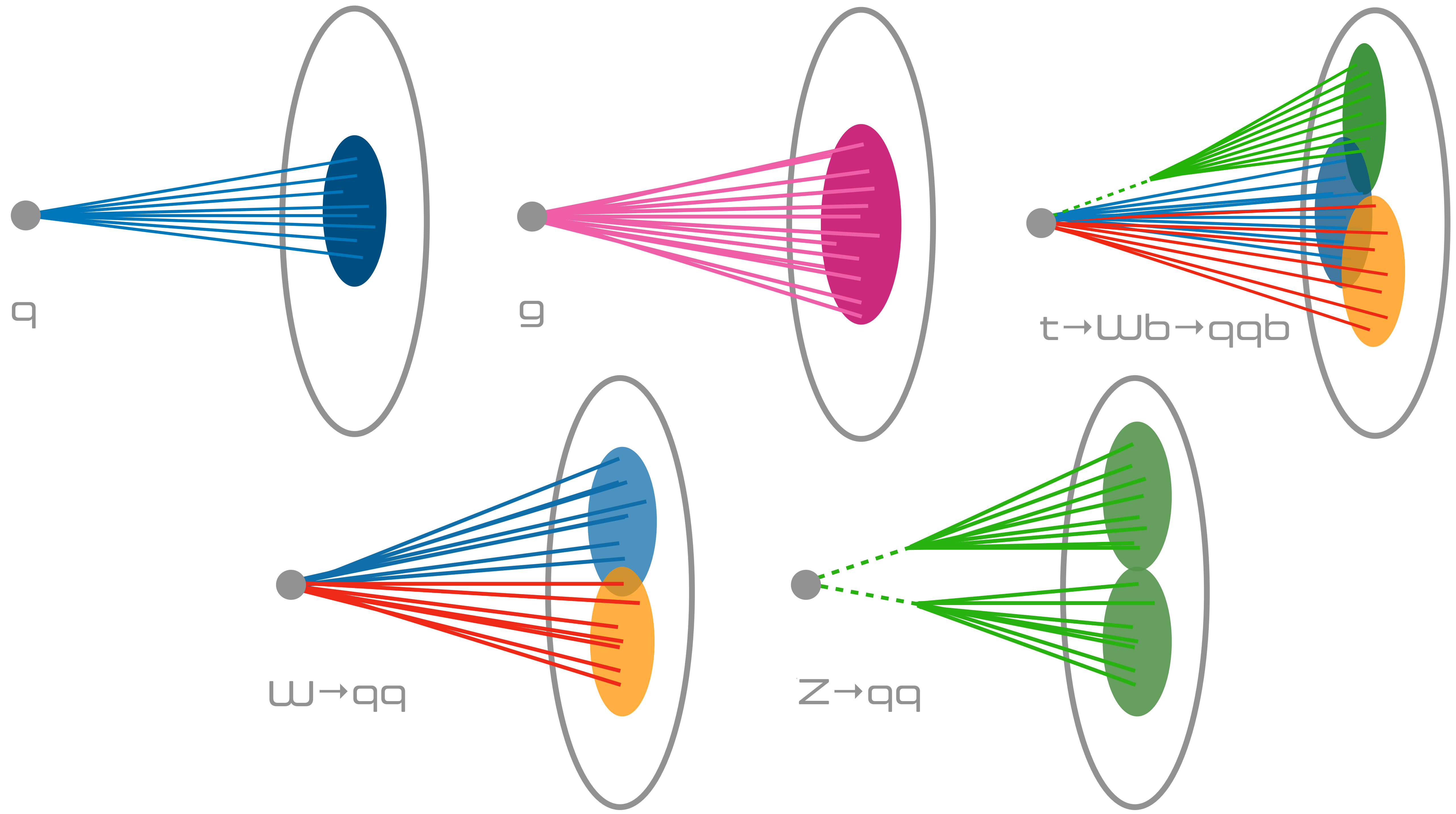
Type	Benchmark	Input Precision	Pipeline Rate	Real-time Latency	Misc. Req.	Baseline Model Parameters
Supervised Learning	Jet Classification	16b	150 ns	1 μs	-	4,389
Unsupervised Learning	Sensor Data Compression	9b	25 ns	100 ns	area, power (65 nm)	2,288
Reinforcement Learning	Beam Control	32b	5 ms	5 ms	-	34,695



CURRENT & FUTURE BENCHMARKS

Type	Benchmark	Input Precision	Pipeline Rate	Real-time Latency	Misc. Req.	Baseline Model Parameters
Supervised Learning	Jet Classification	16b	150 ns	1 μs	-	4,389
Unsupervised Learning	Sensor Data Compression	9b	25 ns	100 ns	area, power (65 nm)	2,288
Reinforcement Learning	Beam Control	32b	5 ms	5 ms	-	34,695

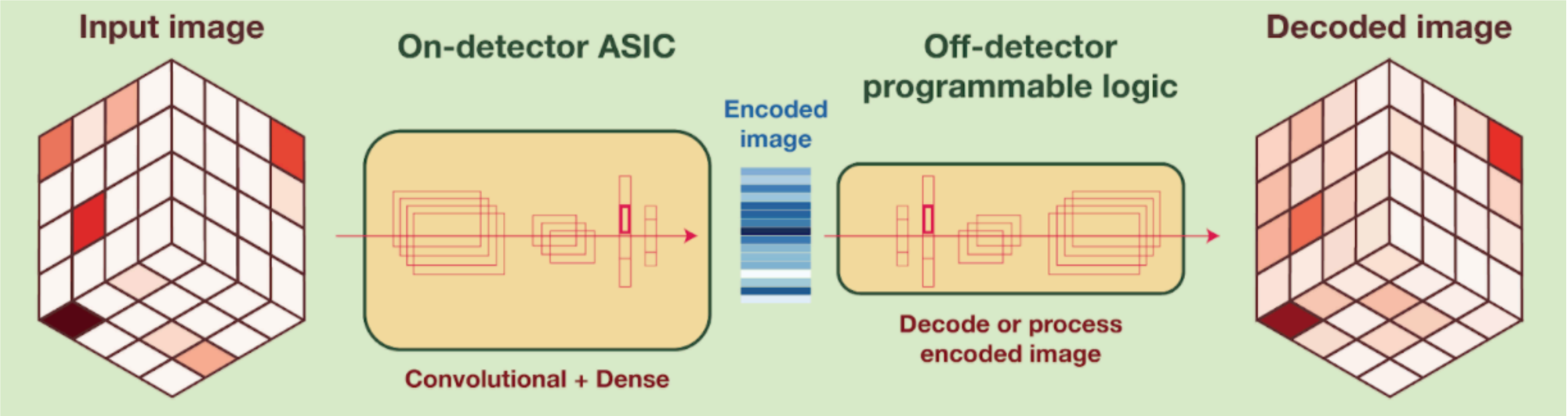
- Particle jet classification for level-1 trigger: ~1 μs latency



CURRENT & FUTURE BENCHMARKS

Type	Benchmark	Input Precision	Pipeline Rate	Real-time Latency	Misc. Req.	Baseline Model Parameters
Supervised Learning	Jet Classification	16b	150 ns	1 μs	-	4,389
Unsupervised Learning	Sensor Data Compression	9b	25 ns	100 ns	area, power (65 nm)	2,288
Reinforcement Learning	Beam Control	32b	5 ms	5 ms	-	34,695

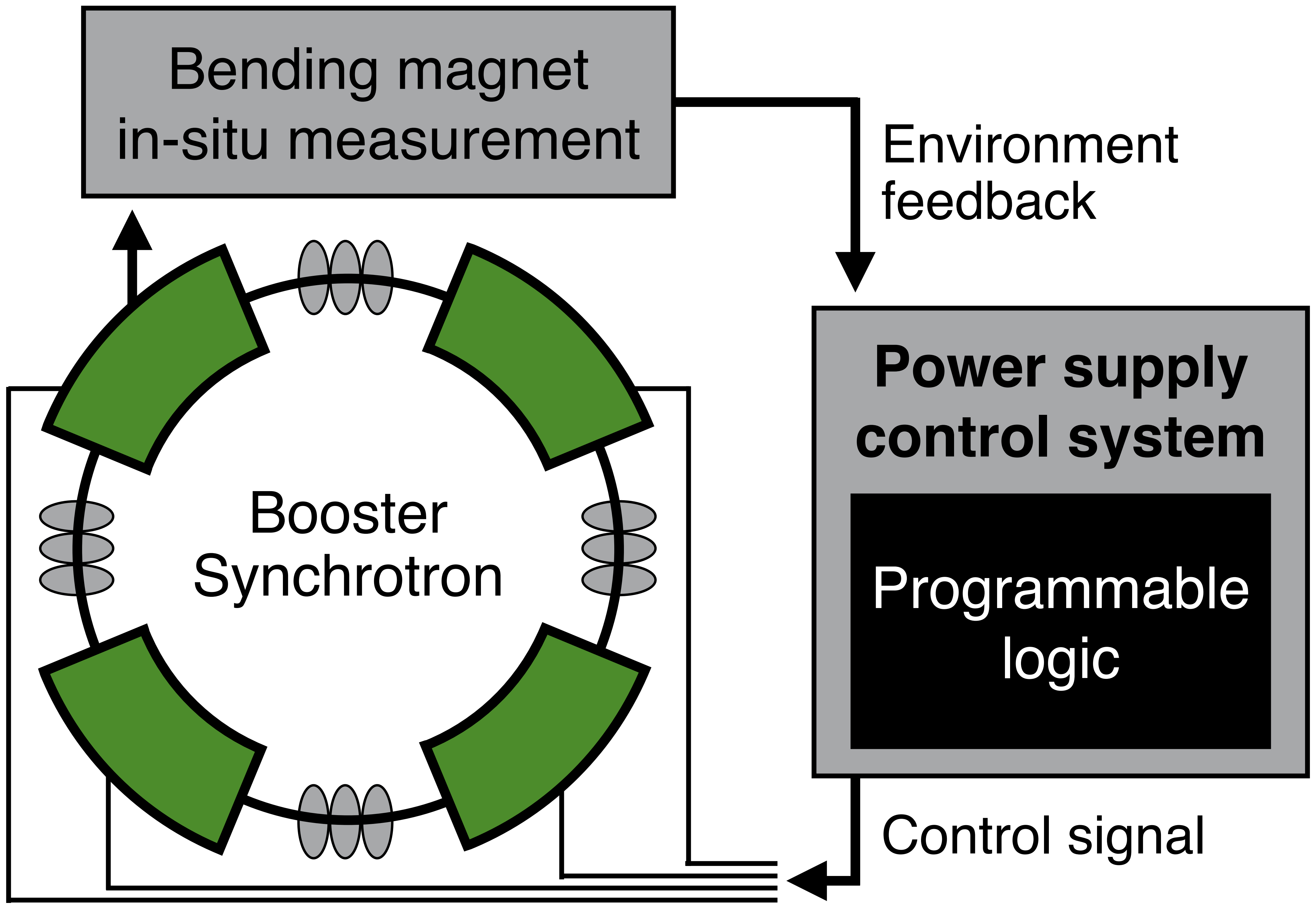
- Particle jet classification for level-1 trigger: ~1 μs latency
- Sensor data compression: ~100 ns latency and additional area/power requirements



CURRENT & FUTURE BENCHMARKS

Type	Benchmark	Input Precision	Pipeline Rate	Real-time Latency	Misc. Req.	Baseline Model Parameters
Supervised Learning	Jet Classification	16b	150 ns	1 μs	-	4,389
Unsupervised Learning	Sensor Data Compression	9b	25 ns	100 ns	area, power (65 nm)	2,288
Reinforcement Learning	Beam Control	32b	5 ms	5 ms	-	34,695

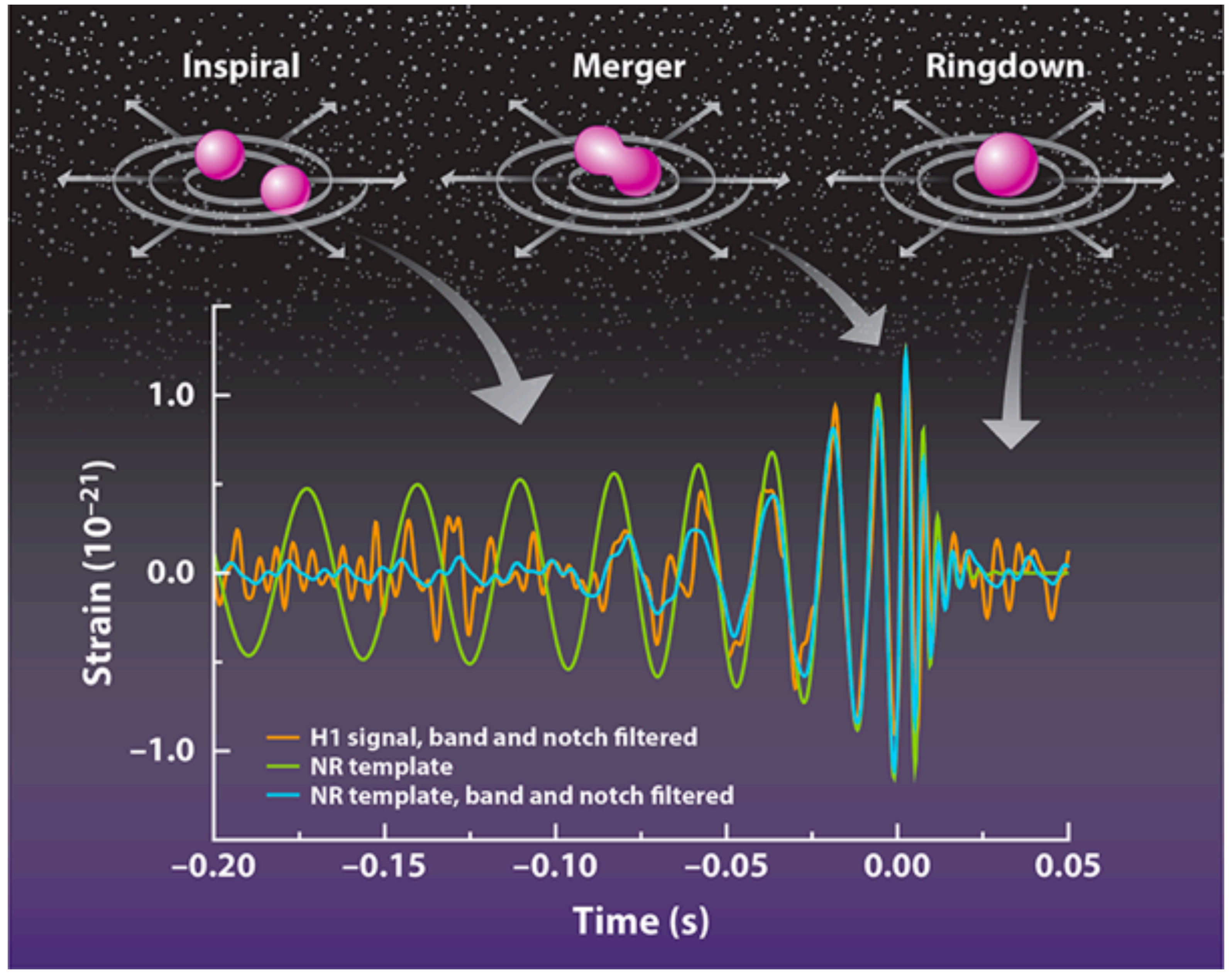
- Particle jet classification for level-1 trigger: ~1 μs latency
- Sensor data compression: ~100 ns latency and additional area/power requirements
- Reinforcement learning for steering accelerator beams: ~5 ms latency

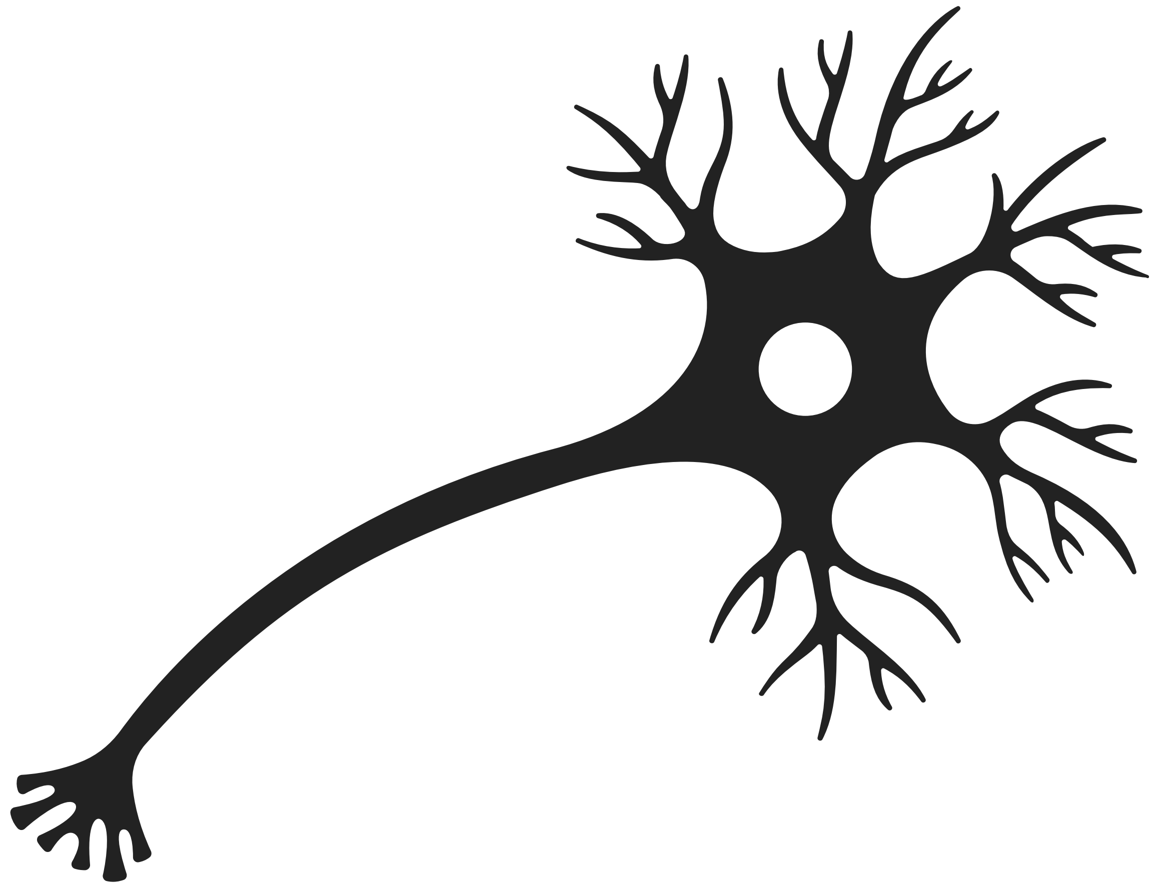


CURRENT & FUTURE BENCHMARKS

Type	Benchmark	Input Precision	Pipeline Rate	Real-time Latency	Misc. Req.	Baseline Model Parameters
Supervised Learning	Jet Classification	16b	150 ns	1 μs	-	4,389
Unsupervised Learning	Sensor Data Compression	9b	25 ns	100 ns	area, power (65 nm)	2,288
Reinforcement Learning	Beam Control	32b	5 ms	5 ms	-	34,695

- Particle jet classification for level-1 trigger: ~1 μs latency
- Sensor data compression: ~100 ns latency and additional area/power requirements
- Reinforcement learning for steering accelerator beams: ~5 ms latency
- *Future: Time sequence analysis for gravitational wave or neural data, and more?*



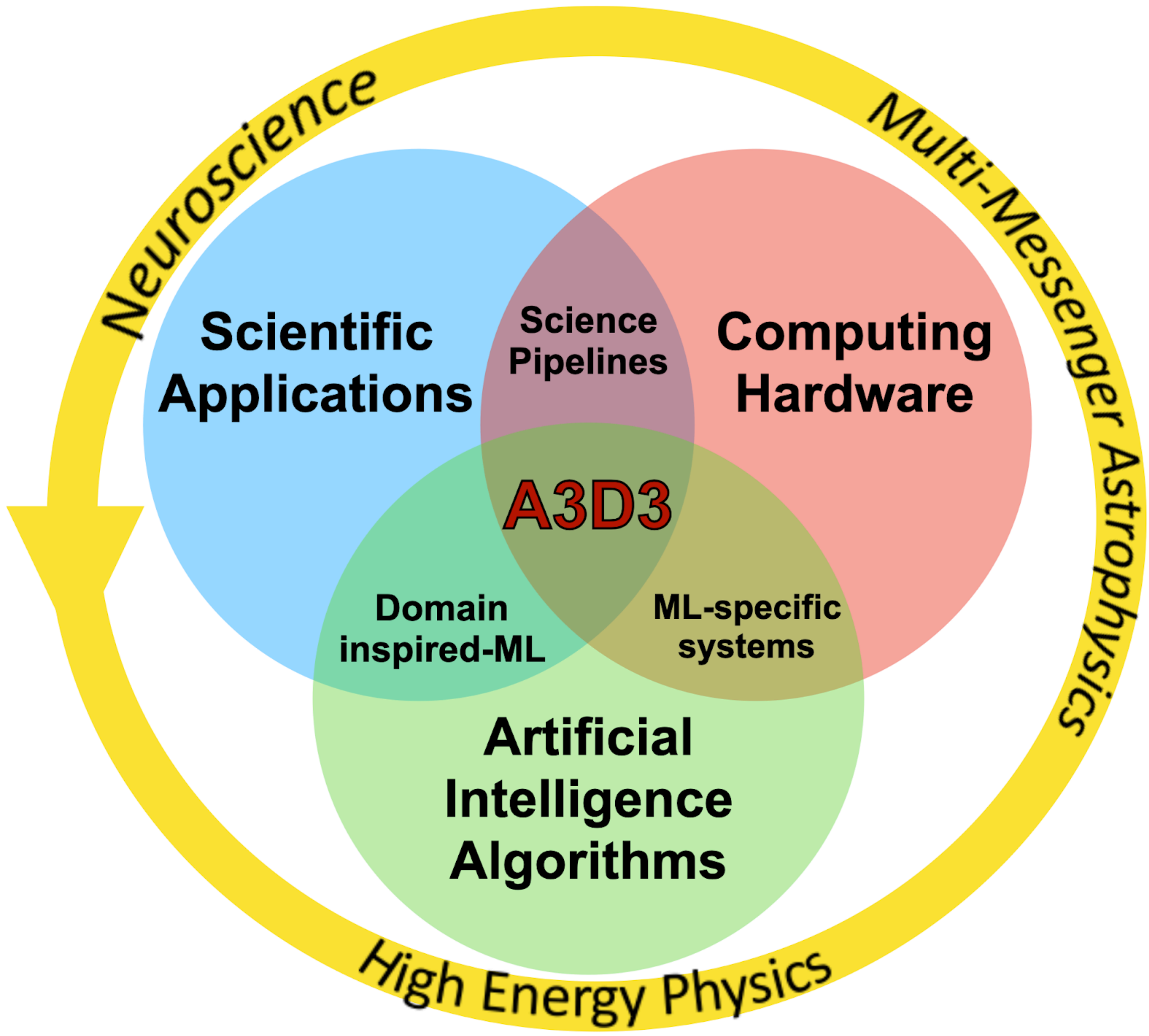


NSF INSTITUTE: A3D3

- Tightly coupled organization of domain scientists, computer scientists, and engineers that unite three core components which are essential to achieve real-time AI to transform science: AI techniques, Computing Hardware, Scientific Applications

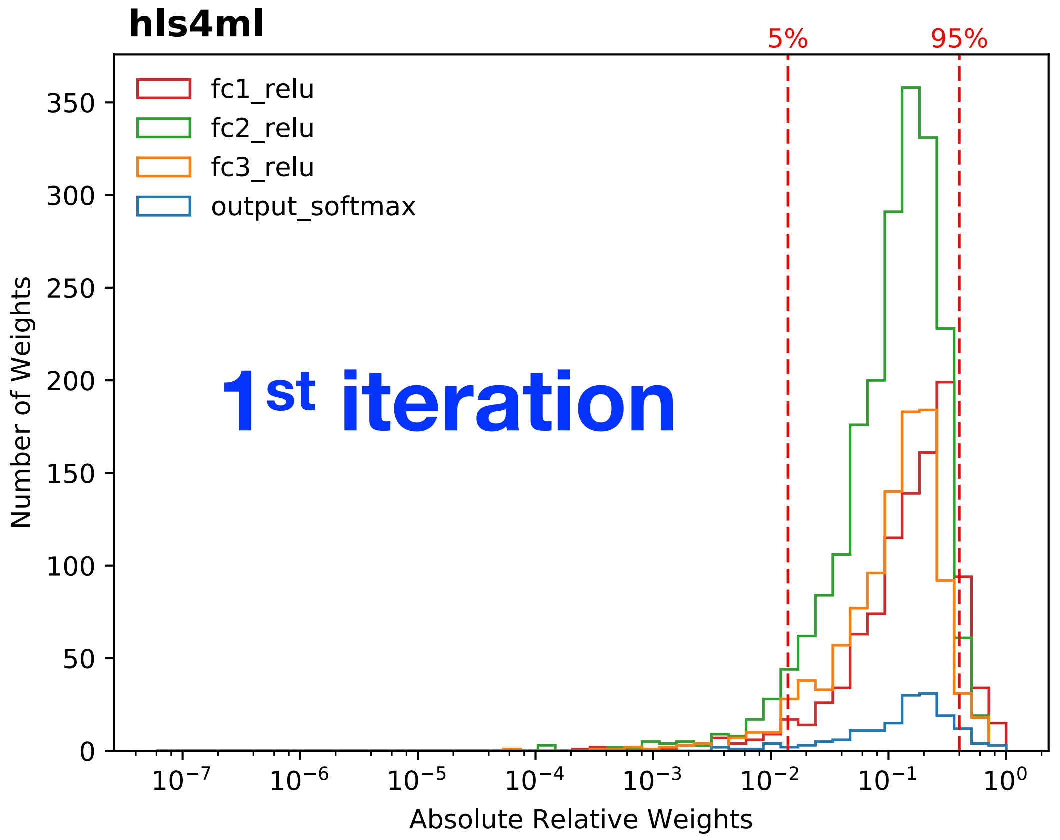
- Collaborators welcome! Check the a3d3.ai for events

OAC-2117997



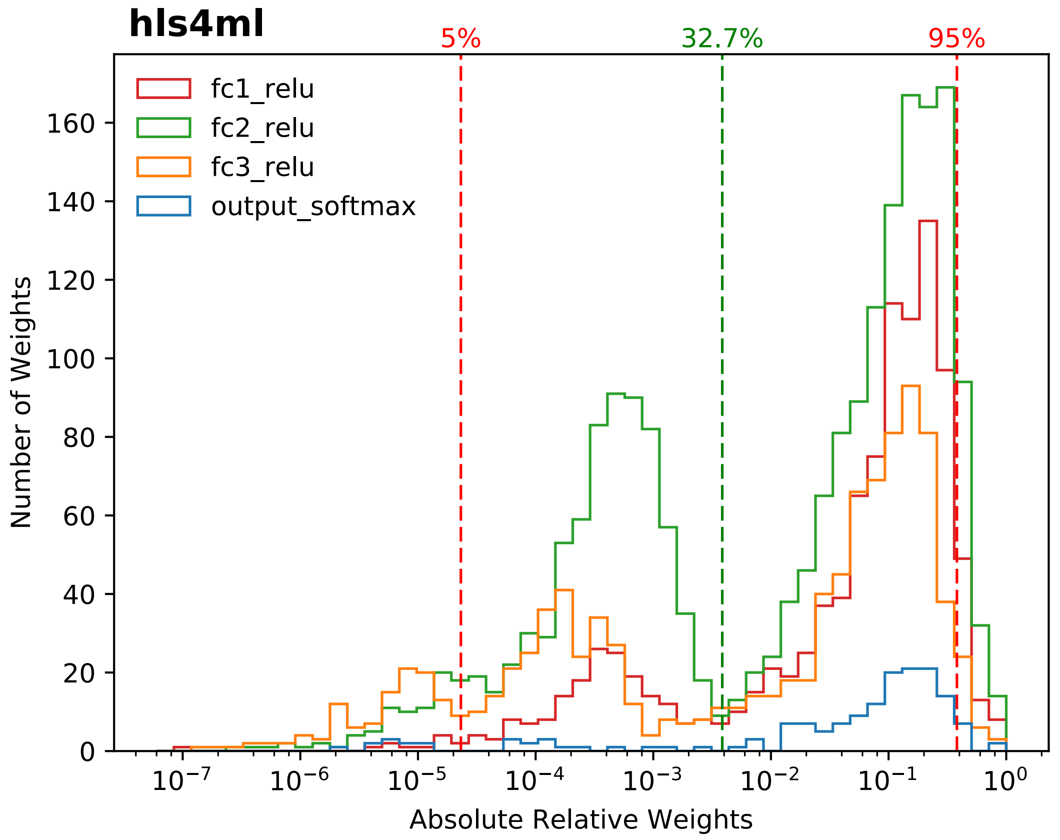
Accelerated AI Algorithms for Data-Driven Discovery

PRUNING

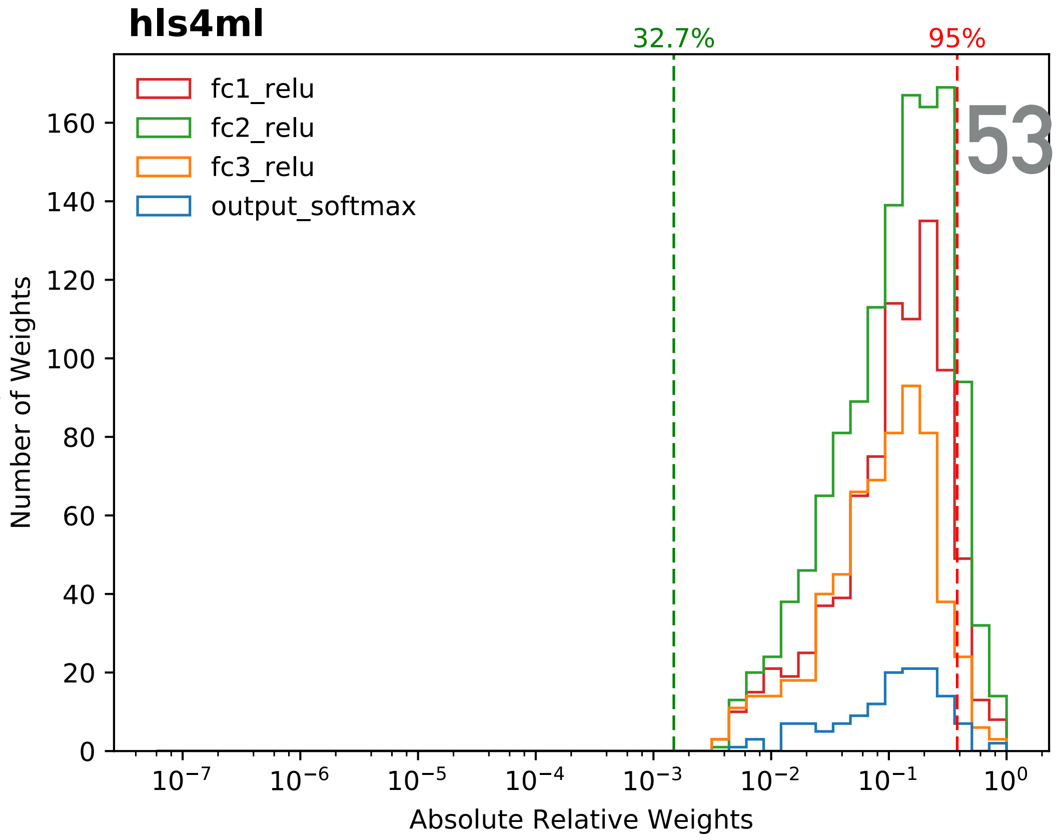


1st iteration

Train with L_1

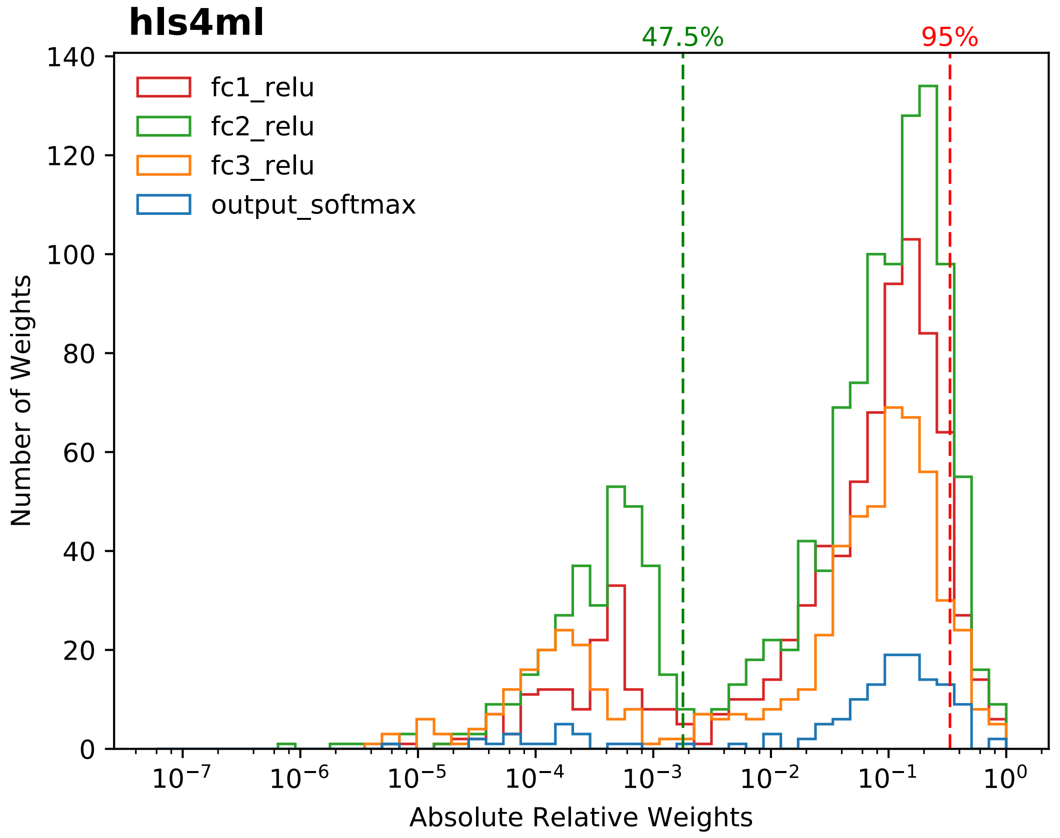


Prune

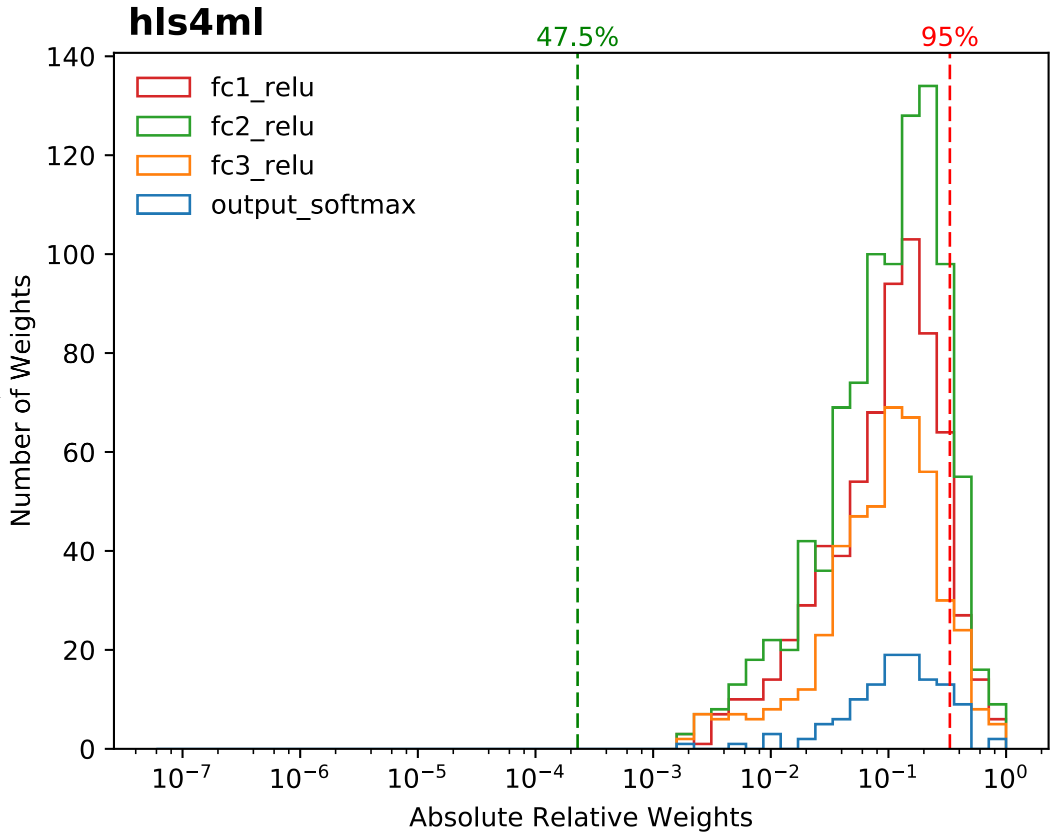


2nd iteration

Retrain with L_1



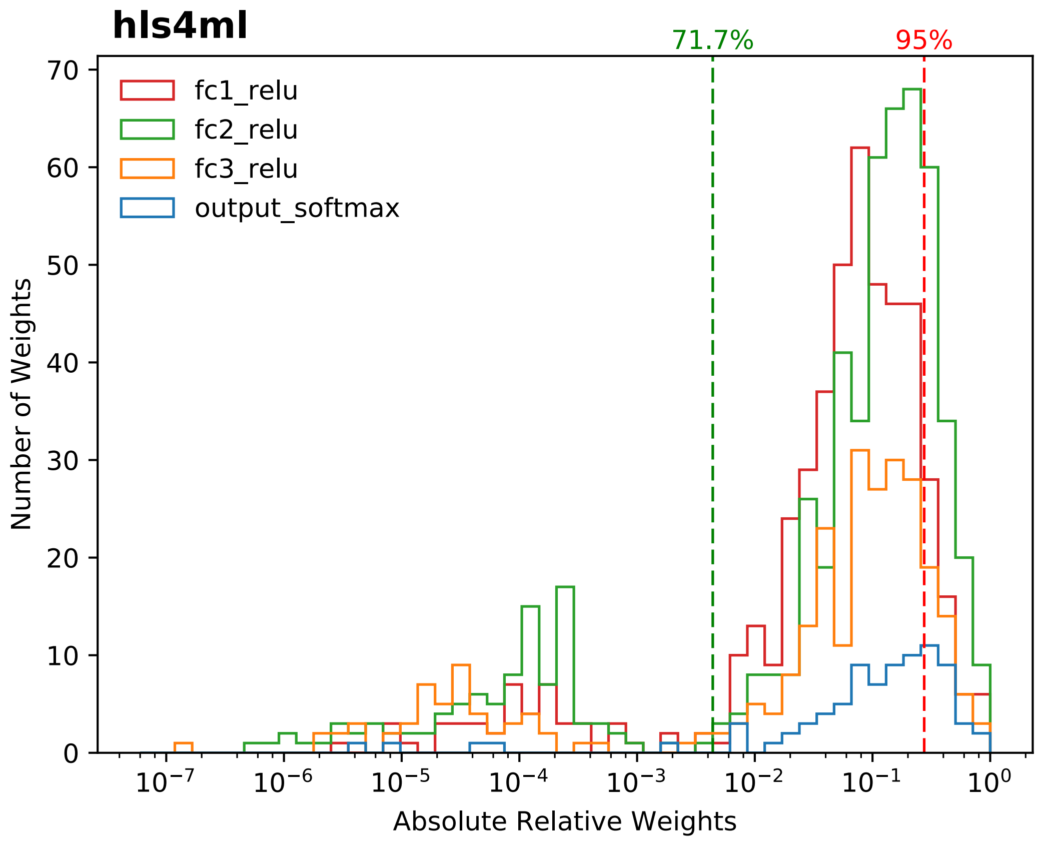
Prune



⋮

7th iteration

Retrain with L_1



Prune